mikroKLAN 9

P. 1877/84

9 1984

informatyka

System ekspertowy
CHILL
Szansa niepełnosprawnych

WYDAWNICTWO NOT SIGMA

informatyka

9 1984

Nr 9
Miesięcznik Rok XIX
Wrzesień 1984

Organ Komitetu Informatyki MNSzWiT oraz Komitetu Naukowo-Technicznego NOT ds. Informatyki

KOLEGIUM REDAKCYJNE:

Mgr inż. Zbigniew GLUZA, dr inż. Wacław ISZKOWSKI, mgr Teresa JABŁOŃSKA (sekretarz redakcji), Władysław KLEPACZ (zastępca redaktora naczelnego), prof. dr hab. Leon ŁUKASZEWICZ (redaktor naczelny), mgr inż. Andrzej J. PIOTROWSKI, dr inż. Janusz ZALEWSKI

STALE WSPÓŁPRACUJĄ:

Mgr inż. Witold ABRAMOWICZ (Szwajcaria), mgr Adam B. EMPACHER, mgr Katarzyna ISAAK, dr Jacek OWCZARCZYK, mgr Marek SOBCZYK, dr Andrzej SZAŁAS, dr Jakub TATARKIEWICZ, mgr inż. Teresa WILCZEK

PRZEWODNICZĄCY RADY PROGRAMOWEJ:

Prof. dr hab. Tadeusz PECHE

Materiałów nie zamówionych redakcja nie zwraca

Redakcja: 00-041 Warszawa, ul. Jasna 14/16, pok. 243 i 244, tel. 27-71-40 lub 26-82-61 w. 184

Zakł. Graf. „Tamka". Zam. 2210. Obj. 4,0 ark. druk. Nakład 4200 egz. T-23.

INDEKS 36124

Cena egzemplarza zł 75,—
Prenumerata roczna zł 900,—

NACZELNA ORGANIZACJA TECHNICZNA
WYDAWNICTWO
CZASOPISM I KSIĄŻEK TECHNICZNYCH
SIGMA
00-950 Warszawa
skrytka pocztowa 1004
ul. Biała 4

W NUMERZE:

	Strona
CHILL — język programowania systemów komutacyjnych (1). Powstanie i założenia języka *Władysław Udrycki*	1
Baza danych obsugiwana przez komputer pośredniczący *Ewa Gutman*	4
Roboty (4). Złożone języki programowania *Cezary Zieliński*	8
MYCIN — wzorcowy system ekspertowy *Oprac. Andrzej Szałas*	11
mikroKLAN	13
— Jaka dokładność numeryczna	
— Elastyczne dyskietki	
— TASWORD II — program redagowania tekstów dla ZX SPECTRUM (48K)	
— Jak zainstalować FORTH (2)	
— Stosowanie kodu maszynowego w komputerach osobistych	
— Zamiana liczb arabskich na rzymskie	
SAMOTESTY	22
— I/C. Buforowanie i blokowanie	
Z KRAJU	24
— Międzynarodowe Targi Książki	
— Rada ds. ZSTO. System bazy danych dla przedsiębiorstw	
ZE ŚWIATA	26
— Komputerowa szansa niepełnosprawnych	
— MACINTOSH	
— Liczyć lepiej	
— AFCET	
TERMINOLOGIA	30
— Nasz drugi język	

W NAJBLIŻSZYCH NUMERACH:

- Roman Żelazny o narzędziach inżynierii oprogramowania
- Jerzy Szyller o mikroprocesorach lat osiemdziesiątych
- Józef de Mezer o sterowaniu alfaskopem
- Uwe Petermann i Andrzej Szałas o przerwaniach
- Emanuel Czyżo i Teresa Latusek o prekursorach informatyki
- Rafał Zieleniewski o problemach programowania mikroprocesorów
- Wiesław Wilczyński o języku PLZ
- Ryszard Rybus o programach uruchomieniowych dla mikroprocesorów
- Ryszard Kott i Krzysztof Szwed o uniwersalnych asemblerach dla mikrokomputerów

WŁADYSŁAW UDRYCKI
Instytut Telekomunikacji
Politechnika Warszawska

P. 1877/84

CHILL — język programowania systemów komutacyjnych (1)

Powstanie i założenia języka

W telekomunikacji, podobnie jak w informatyce, od wczesnych lat sześćdziesiątych projektanci sytemów dążą do doskonalenia metod i narzędzi stosowanych w różnych fazach tworzenia oprogramowania. Opracowywane są metody oprogramowania systemów komutacyjnych i języki ich opisu funkcjonalnego. Powstają wyspecjalizowane języki, odpowiednie do potrzeb programowania w telekomutacji, oraz języki dialogu z systemem komutacyjnym.

Wraz z rozwojem systemów komutacyjnych sterowanych programowo, wzrósł stopień komplikacji związanego z nimi oprogramowania. Wielkość złożonego funkcjonalnie oprogramowania współczesnych systemów komutacyjnych dochodzi do miliona instrukcji. Jego część sterująca obsługuje znaczną liczbę procesów współbieżnych o silnych uzależnieniach czasowych. Dla przykładu — w centrali o 50 tysiącach abonentów przeciętnie obsługiwanych jest około 5 tysięcy połączeń jednocześnie. Równolegle z zadaniami sterowania, wykonywane są programy utrzymaniowe (ang. maintenance) i administracyjne — kilkakrotnie większe od oprogramowania sterującego.

Oprogramowanie komutacyjne powinno zapewniać łatwą adaptację do potrzeb zróżnicowanych funkcjonalnie central. Jednocześnie musi być ono podatne na wprowadzanie zmian związanych z modyfikacją i rozszerzaniem w okresie „życia" systemu komutacyjnego — od dwudziestu do czterdziestu lat. W tak długim czasie zmieniają się bowiem wymagania funkcjonalne ze strony otoczenia. Wymaga się, by zmiany oprogramowania były wprowadzane na bieżąco, bez przerw w pracy systemu.

Poza tym oprogramowanie komutacyjne musi charakteryzować szczególna niezawodność. Jest to jedno z najistotniejszych wymagań stawianych tym systemom, bowiem wiąże się ono z wymaganiem ciągłości i pewności pracy służb telekomunikacyjnych. Przyjmuje się, że system komutacyjny powinien dokonywać 99,98% poprawnych połączeń, a sumaryczny czas przerw w działaniu, w ciągu czterdziestu lat istnienia systemu, nie powinien przekroczyć 2 godzin. Zatem oprogramowanie komutacyjne musi mieć zdolność działania m i m o błędów powstałych z winy sprzętu bądź — defektów programowych.

Wyspecjalizowane języki do programowania systemów komutacyjnych, ogłoszone w latach 1970—1974

Język	Rok publikacji	Ośrodek
TPL1	1970	University of Essex, Wielka Brytania
TPL2	1972	UK Post Office, Wielka Brytania
TPL1.5	1973	University of Essex, Wielka Brytania
ESPL/1	1973	ITT
TSPL	1973	GTE, USA
COMSYL II	1973	Bell Telephone Lab., USA
EPL	1974	Bell Telephone Lab., USA
PAPE	1974	CNET, Francja
DPL	1974	NTT, Japonia
PLEX	1974	Ellemetel, Szewcja

Źródło: Hills M.T., Kano S.: Programming electronic switching systems. Peter Peregrinus Ltd., 1976

Dla sprostania potrzebom programowania systemów komutacyjnych opracowano wiele wyspecjalizowanych języków programowania. Od 1970 do 1974 roku opublikowano informacje o dziesięciu językach wysokiego poziomu, opracowanych na Uniwersytecie w Essex i ośrodkach badawczych różnych firm telekomunikacyjnych (p. tab.). Używane były także niektóre języki uniwersalne, m.in. PASCAL i MARY.

PRACE CCITT I POWSTANIE CHILLA

Od końca lat sześćdziesiątych problem programowania systemów komutacyjnych jest przedmiotem zainteresowania CCITT (Comité Consultatif International Télégraphique et Téléphonique) — Międzynarodowej Unii Telekomunikacji w Genewie. Już w 1968 roku wstępnie dyskutowano problemy oprogramowania komutacyjnego, podkreślając jego wielkość i złożoność funkcjonalną, potrzebę szczególnej niezawodności (pewności działania i odporności na błędy) oraz potrzebę elastyczności w operowaniu oprogramowaniem. Ustalono konieczność stworzenia wyspecjalizowanego języka wysokiego poziomu, który sprostałby wszystkim wymaganiom programowania w telekomutacji. Jednocześnie — ze względu na zainteresowanie wielu firm rozwojem oprogramowania komutacyjnego — językowi takiemu nadano rangę standardowego środka komunikacji projektantów (producentów).

Programiści systemów komutacyjnych bardzo wcześnie rozróżnili fazy: projektowania, programowania i użytkowania oprogramowania. Uwzględniając zróżnicowane potrzeby wyodrębnionych faz, podjęto decyzję opracowania trzech języków:

- **SDL** (Specification and Description Language) — język specyfikacji i opisu funkcjonalnego
- **CHILL** (CCITT High Level Language) — język wysokiego poziomu do programowania systemów o działaniu współbieżnym
- **MML** (Man Machine Language) — interakcyjny język dialogu człowieka z systemem komutacyjnym.

Według intencji twórców języków SDL jest przeznaczony do stosowania głównie w fazie projektowania oprogramowania, CHILL do jego realizacji, a MML do eksploatacji gotowego systemu.

Opracowane w latach 1968—1972 założenia dla tych języków przyjęto na zebraniu plenarnym CCITT (1972), na którym utworzono trzy grupy specjalistów. Odtąd prace prowadzono równolegle, a ich wynik otrzymał rangę „Zaleceń CCITT", zatwierdzonych w Genewie w 1980 roku.

Prace nad językiem CHILL rozpoczęto od oceny 27 języków zaproponowanych do standaryzacji, w efekcie ograniczając ich liczbę do siedmiu. Jednakże wybór jednego z nich okazał się niemożliwy, gdyż zdania były podzielone między zwolenników języków ALGOL, PASCAL i PL/I. W tej sytuacji podjęto decyzję opracowania nowego języka, nie wzorowanego bezpośrednio na żadnym z wymienionych. Pierwsza propozycja powstała w roku 1976, po czym kontynuowano prace nad rozwinięciem defnicji języka oraz oceną jego praktycznego zastosowania. Następna propozycja przedstawiona w 1979, została — po wprowadzeniu pewnych poprawek — zatwierdzona w roku 1980 i opublikowana jako Zalecenie CCITT Z200 [1]. Ponadto wydano dwa opracowania: „Wprowadzenie do CHILLA" [2] oraz „Opis formalny CHILLA" [3].

Opracowanie CHILLA przez liczną grupę projektantów trwało pięć lat i kosztowało kilkadziesiąt milionów dolarów. Prace nad językiem są nadal kontynuowane. Od 1981 roku prowadzi się studia nad specyfikacją środowiska języka i jego zastosowaniem praktycznym. Język został dobrze przyjęty przez programistów, o czym świadczy fakt, iż już w roku 1981 dziewięć światowych firm telekomunikacyjnych zbudowało kompilatory CHILLA, a jedenaście innych było w trakcie ich tworzenia [4].

ZAŁOŻENIA I ZAKRES ZASTOSOWANIA

CHILL opracowywano przy założeniu następujących wymagań sformułowanych przez CCITT:

— zwiększenie niezawodności oprogramowania przez wnikliwe badanie programów w czasie kompilacji

— generowanie wysokoefektywnego programu wynikowego

— duża elastyczność i uniwersalność, umożliwiające użycie w ramach wielu dziedzin zastosowań, a także wykorzystanie różnego rodzaju sprzętu

— ułatwienie modularnego i strukturalnego programowania

— łatwość uczenia się i stosowania.

Choć CHILL projektowano przede wszystkim pod kątem programowania systemów komutacyjnych, uważa się, iż jest on wystarczająco uniwersalny do programowania szerokiej klasy systemów o działaniu współbieżnym. Przewiduje się, że CHILL będzie służył jako język zaspokajający wiele potrzeb związanych z oprogramowaniem systemów komutacyjnych. Poza użyciem w oprogramowaniu sterującym, może być stosowany do tworzenia oprogramowania utrzymaniowego, administracyjnego oraz systemów operacyjnych. Przewidywane jest także zastosowanie CHILLA do modelowania przy projektowaniu systemów współbieżnych.

CHILL jest językiem maszynowo niezależnym. Jednakże, dla zwiększenia efektywności programu wynikowego, może on zawierać definiowane podczas implementacji operacje niskiego poziomu i mieć możliwość współpracy z poziomem asemblera. Potrzeby takie istnieją podczas programowania współpracy ze sprzętem — na przykład, gdy CHILL jest używany do tworzenia systemu operacyjnego.

Ze względu na długi okres „życia" systemu komutacyjnego, CHILL zaprojektowano jako język samodokumentacyjny. Dla umożliwienia modyfikacji oprogramowania w ciągu tego okresu, CHILL pomyślany jest jako narzędzie opisu logiki programu. Dzięki właściwościom języka, struktura oprogramowania i jego treść mogą być czytelne i zrozumiałe bez dodatkowych informacji.

CHILL A NIEZAWODNOŚĆ OPROGRAMOWANIA

CHILL wyposażono w nowoczesne środki zwiększania niezawodności stosowane zarówno podczas tworzenia, jak i wykonywania programów. Dla zwiększenia niezawodności programów z jednoczesnym zachowaniem ich efektywności, rozbudowano szczególnie środki stosowane przy pisaniu oraz uruchamianiu programów. W celu zwiększenia niezawodności związanej z fazą pisania programów wprowadzono konstrukcje językowe pozwalające na unikanie błędów. Konstrukcje te — z jednej strony — umożliwiają pisanie programów w sposób zbliżony do naturalnego opisu narracyjnego, z drugiej zaś — przybliżają proces programowania do myślenia konstrukcyjno-projektowego, dzięki modularyzacji i hierarchicznej strukturalizacji oprogramowania.

Dwa rodzaje jednostek programowych wywoływanych przez nazwę — **procedury** i **procesy** — są głównymi środkami strukturalizacji oprogramowania w CHILLU. Rozbudowane możliwości stosowania procedur i przekazywania parametrów, ich wielowejściowość i wielowyjściowość, możliwość zagnieżdżania — ułatwiają budowę hierarchicznego oprogramowania strukturalnego. Możliwości te wsparte są dużą różnorodnością instrukcji strukturalnych. Procesy, podobne w budowie do procedur, są specjalnymi konstrukcjami syntaktycznymi służącymi do programowania współbieżnego.

W fazie uruchamiania programu, bardzo istotna — z punktu widzenia niezawodności — jest efektywność usuwania błędów podczas kompilacji. Wykorzystywane w CHILLU środki temu służące — to separacja danych z określeniem reguł ich zasięgu (ang. scope), oraz typizacja danych z regułami ustalania zgodności typów. Zastosowano pięć konstrukcji izolujących jednostki programu wraz z obiektami danych: **moduły, rejony, procedury, procesy** i **bloki begin.**

W CHILLU operuje się pojęciami okresu istnienia (ang. life time) zmiennych i widoczności (ang. visibility) nazwy. Związane z nimi reguły stalone dla pojedynczych i zagnieżdżonych jednostek programu, umożliwiają wnikliwą kontrolę użycia zmiennych. Reguły te określają również uprawnienia programisty, któremu zaoferowano możliwości świadomego zmieniania okresu istnienia zmiennych i widoczności nazw.

Metody polegające na unikaniu błędów oraz ich wykrywaniu przez kompilator pozwalają na tzw. statyczne usuwanie błędów. W CHILLU zawarto również obsługę pewnych sytuacji związanych z uwarunkowaniami, które nie mogą być kontrolowane statycznie. Przekroczenie tych uwarunkowań (ang. run time exception) powoduje wstrzymanie działania programu oraz przekazanie sterowania do specjalnego programu obsługi. CHILL określa podstawowy zbiór dynamicznych uwarunkowań, przewidując możliwość definiowania dodatkowych podczas implementacji języka lub budowy oprogramowania użytkowego. Zbiór ten może więc uwzględniać uwarunkowania narzucone przez projektantów implementacji języka, z uwzględnieniem np. wymagań maszyny i jej otoczenia, a także — przez programistów. Ci ostatni mogą tworzyć specjalne programy,obsługujące dynamicznie uwarunkowane koincydencje zdarzeń, które mogłyby spowodować niepoprawne zachowanie się oprogramowania.

ZARYS JĘZYKA

Dostępnymi dla programisty komponentami programów w CHILLU są **obiekty danych, akcje** (ang. actions) i **struktura programu.** Język umożliwia dokonywanie:

— **opisu obiektów danych** — za pomocą deklaracji i definicji

— **opisu akcji** — określającego jakie operacje mają być dokonywane na obiektach danych oraz kolejność ich wykonywania

— **opisu struktury programu** — ustalającego podział programów i dostępność obiektów danych.

Program w CHILLU jest sekwencją **modułów** i **rejonów** otoczonych definicją procesu zewnętrznego. W zasięgu tego procesu istnieje moduł definicji nazw języka (implementacji) i jego procedur standardowych.

Moduły są najbardziej zewnętrzną konstrukcją nawiasową, służącą do podziału funkcjonalnego oprogramowania. Umożliwiają one rozłączną kompilację programów. Każdy moduł zawiera lokalne deklaracje i definicje oraz ciąg instrukcji. Moduły mogą być zagnieżdżane.

Rejon jest szczególnym rodzajem modułu umożliwiającego koordynację dostępu do zadeklarowanych w nim wspólnych zasobów. Rejon może zawierać tylko definicje i deklaracje zmiennych — nie jest więc programem. Zdefiniowane w nim procedury umożliwiają konkurencyjny dostęp do zasobów, o które ubiegają się różne procesy. Rejony nie mogą być zagnieżdżane w innych rejonach i modułach. Procedury rejonu nie mogą wywoływać procedur innego rejonu — ani bezpośrednio, ani pośrednio (tj. przez inne procedury).

Moduł, rejon, procedura, proces i **blok begin** ograniczają widoczność nazw i okres istnienia zmiennych, uniemożliwiając ich nielegalne użycie. Okresem istnienia zmiennej określa się przedział czasu związany z jej alokacją. Nazwa jest widoczna w określonym punkcie programu, jeżeli może być w nim użyta. Zasięg nazwy wskazuje wszystkie punkty, w których jest ona widoczna.

Moduły i **rejony** ograniczają tylko widoczność nazw i nie mają wpływu na okres istnienia zmiennych. Nazwy deklarowane wewnątrz modułu traktowane są jako lokalne. Nazwy globalne nie są automatycznie widoczne wewnątrz modułu. Jednakże istnieje możliwość świadomej zmiany przez programistę widoczności do wewnątrz i na zewnątrz modułu.

Procedura, proces i **blok begin,** nazywane w oryginale opisu **blokami,** mają jednakowe reguły zasięgu. Zmienne deklarowane wewnątrz bloku istnieją tylko w jego granicach i widoczne są we wszystkich blokach wewnętrznych.

CHILL jest językiem o silnej typizacji obiektów danych. Określane są one za pomocą typu i atrybutów ujmujących takie właściwości, jak: struktura, wymiar, uporządkowanie, zbiór dozwolonych wartości, zbiór dozwolonych działań, sposób odwoływania się, ograniczenie dostępności tylko do odczytu. Typy danych, utworzone na podstawie tych właściwości, można pogrupować w następujące kategorie:

typ prosty — zmienne całkowite, znakowe i logiczne oraz typ wyliczeniowy i okrojony

typ złożony — ciągi bitowe i znakowe, tablice, struktury

typ zbiorowy — zbiory elementów prostych

typ wskaźnikowy — wskaźniki swobodne, związane i dynamiczne

typ procedurowy — procedury traktowane jako manipulowane obiekty danych

typ identyfikacyjny — identyfikatory procesów

typ synchronizacyjny — zdarzenia, bofory i sygnały służące do synchronizacji i komuniakcji procesów.

Dla zwiększenia elastyczności struktur danych i efektywności operowania nimi, CHILL uzupełniono o pewne cechy języka niskiego poziomu, stwarzając możliwości zadawania sposobu rozmieszczenia (ang. layout) danych. Z drugiej strony — programista może rozszerzyć zawarty w CHILLU zestaw typów danych, definiując nowy typ za pomocą tych, które już istnieją w języku. Oprócz tego, CHILL umożliwia stosowanie dynamicznych obiektów parametrycznych tj. takich, których przynajmniej jeden parametr jest ustalony podczas wykonywania programu. Istnieje również możliwość stosowania wskaźników do tego rodzaju obiektów.

Algorytmiczną część programu w CHILLU tworzy ciąg instrukcji — składający się na opis podejmowanych akcji. CHILL charakteryzuje się dużą różnorodnością instrukcji, co jest cechą nowoczesnych języków programowania. Zestaw instrukcji w CHILLU jest dostosowany zarówno do uniwersalnych zadań programowania sekwencyjnego, jak i do programowania współbieżnego. Instrukcje te można podzielić na następujące grupy: proste, strukturalne, operowania procedurami i sterowania procesami.

Instrukcje proste umożliwiają wykonywanie podstawowych działań przypisania i skoku. Przypisanie wartości zmiennym w CHILLU może być dokonywane dla zmiennych prostych i złożonych. Instrukcją przypisania można nadać wartość całej zmiennej złożonej lub jej części, np. dowolnym przekrojom tablicy czy wydzielonym polom rekordu.

Zbiór instrukcji strukturalnych w CHILLU, zgodnie z tendencją strukturalizacji programów, zawiera bogaty zestaw urozmaiconych instrukcji warunkowych i iteracyjnych. CHILL umożliwia oprogramowanie złożonych warunków IF i CASE, łącznie z użyciem tablic decyzyjnych Instrukcje warunkowe mogą być zagnieżdżane, przy czym oferowane konstrukcje umożliwiają zmniejszenie stopnia zagnieżdżania celem zwiększenia czytelności programu.

Instrukcje iteracyjne wraz z częścią sterującą FOR i WHILE umożliwiają tworzenie różnych wariantów pętli ułatwiających programowanie. Istnieje możliwość budowania pętli nieskończonych, w przypadku których szczególnie ważne są możliwości niestandardowego opuszczania pętli.

Procedury w CHILLU mogą być wielowejściowe i wielowyjściowe. Punkty wejściowe są zaznaczone w treści procedury i opatrzone nazwą. Opuszczanie procedury następuje po zakończeniu ciągu instrukcji wyznaczonego jej definicją lub w wyznaczonych przez programistę miejscach związanych np. z warunkiem określonym w procedurze.

Procedury mogą być używane dynamicznie, tzn. przekazywane jako parametr do innej procedury, przesyłane jako komunikat do **bufora**, podstawiane jako wartość zmiennej, itp. Wywołanie procedury może odbywać się z przeniesieniem wartości lub zmiennych. Obydwa sposoby są stosowane zarówno dla przekazywania parametrów wejściowych, jak i wyjściowych, tj. dostarczonych przez procedurę. Parametry formalne mogą mieć atrybuty określające kierunek ich przekazywania. W CHILLU przewidziano również możliwość wyznaczania rejestrów do przekazywania wartości przez procedury.

Typ procedury oraz typ przekazywanych parametrów są deklarowane w definicji procedury. Niezależnie od tego, procedura może być wyposażona w atrybuty określające jej właściwości i sposób wykorzystania w programie. Atrybuty określają rekursywność bądź nierekursywność procedury, możliwość bądź zakaz dynamicznego jej używania itp.

CHILL zaprojektowano jako język programowania współbieżnego. Podstawowymi konstrukcjami umożliwiającymi to programowanie są **procesy i rejony.**

Koncepcję **procesu** stworzono specjalnie po to, aby umożliwić równoległe wykonywanie fragmentów oprogramowania oraz dynamiczne uzależnienie sposobu wykonywania każdego z fragmentów od reakcji jego otoczenia. Program w CHILLU zawiera **definicje procesów** traktowane jako wzorce, z których tworzone są procesy. Nowy proces może być wyzwolony przez inny proces w wyniku zainicjowania programu objętego jego definicją i jest uważany za współbieżny z procesem, który go wyzwolił. Zasady CHILLA dopuszczają równoczesne wykonywanie wielu procesów związanych z jedną lub większą liczbą definicji. Każdy z tych procesów ma identyfikator o wartości nadawanej w chwili jego wyzwolenia. Proces, który został wyzwolony, a nie jest zakończony, znajduje się zawsze w jednym z dwu możliwych stanów **aktywnym** (ang. active) lub **oczekiwania** (ang. delayed). Zmiany stanu następują w wyniku dyktowanych przez program działań związanych z metodami koordynacji procesów.

Zawarte w CHILLU środki koordynacji procesów można rozpatrywać w trzech aspektach:

- **synchronizacji** — dokonywanej przez wstrzymywanie wykonywania procesów (opóźnianie) do czasu wystąpienia warunków umożliwiających wznowienie ich wykonywania
- **komunikacji** — następującej wskutek przesyłania komunikatów tj. obiektów danych między procesami
- **wzajemnego wykluczania** — umożliwiającego konkurencyjny dostęp procesów do ich wspólnych zasobów.

Środkami synchronizacji procesów są **zdarzenia** (ang. events) Proces może być w stanie oczekiwania na zajście wytworzonego przez inny proces zdarzenia, które spowoduje wznowienie jego wykonywania. Synchronizacja i komunikacja procesów jest dokonywana za pomocą **sygnałów** i **buforów.** Komunikaty przekazywane za ich pośrednictwem są przechowywane do czasu odebrania przez właściwy proces odbiorczy.

Blok programowy — **rejon** — jest stosowany do organizowania wzajemnego wykluczania dostępu do wspólnych zasobów. Obszary krytyczne danych są deklarowane w rejonie jako lokalne i mogą być osiągane tylko przez procedury rejonu, to inny proces wywołujący tę lub inną procedurę tego rejonu zostaje opóźniany aż do zakończenia pracy wywołanej procedury. Zatem związany z rejonem mechanizm blokowania dostępu do wspólnych zasobów pozwala na korzystanie z nich tylko jednemu procesowi, ustawiając pozostałe w stan oczekiwania.

cdn.

LITERATURA

[1] CCITT: CHILL Language Definition. CCITT Recommendation Z200, November 1980

[2] CCITT: Introduction to CHILL. CCITT Manual, May 1980

[3] CCITT: Formal Definition of CHILL. CCITT Study Group XI Implementors Form, December 1981

[4] Rekdal K.: CHILL — The Standard Language for Programming SPC Systems. IEEE Transactions on Communications, June 1982.

SPROSTOWANIE

W numerze 7/84 na dwóch stronach (1 i 2) trzykrotnie podaliśmy błędną pisownię nazwiska **Gary Kildall.** Autorów, Czytelników oraz p. Kildalla serdecznie przepraszamy. (Red.)

EWA GUTMAN
CPiZI ZETO ZOWAR
Warszawa

Baza danych obsługiwana przez komputer pośredniczący

Charles Bachman, otrzymawszy w 1973 roku nagrodę Turinga, wygłosił wykład, w którym przedstawił rysującą się zmianę istoty zjawiska zwanego przetwarzaniem danych. Dotychczas uważano, że centralnym elementem tego przetwarzania jest komputer, a dane są jedynie strumieniem płynącym przez maszynę. Obecnie coraz powszechniejszy staje się pogląd, że podstawą przetwarzania jest baza danych, a komputer jest traktowany tylko jako narzędzie służące do jej obsługi [2].

Idea wyspecjalizowanej maszyny obsługującej wyłącznie bazę danych i wspomagającej komputer centralny bynajmniej nie narodziła się ze zbyt dosłownego zrozumienia przedstawionej powyżej koncepcji przetwarzania danych. Pomysł odciążenia komputera centralnego polegający na przekazaniu niektórych jego funkcji wyspecjalizowanemu minikomputerowi został wykorzystany jeszcze w latach sześćdziesiątych, co prawda nie w dziedzinie baz danych, ale w teleprzetwarzaniu. W rozwiązaniu tym wszystkie funkcje komputera centralnego, związane z komunikacją z terminalami, zostały przeniesione na minikomputer, pełniący funkcję swoistej „kancelarii" systemu. Takie rozwiązanie konfiguracji systemu komputerowego nazwane zostało „front-end configuration", a wyspecjalizowany minikomputer — „front-end processor".

Wraz ze wzrostem zastosowań dużych baz danych, problem przeciążenia komputera centralnego stał się na tyle istotny, że przed wieloma użytkownikami pojawiła się alternatywa: albo zakupić komputer o większej mocy obliczeniowej, albo znacznie ograniczyć zakres zastosowań. W tej sytuacji pomysł przeniesienia części oprogramowania na minikomputer wydał się dość atrakcyjny, zwłaszcza wobec imponujących ofert rynku mini- i mikrokomputerowego.

ARCHITEKTURA SYSTEMU I SCHEMAT DZIAŁANIA

Pierwszą próbą wykonania konfiguracji wykorzystującej osobny komputer do obsługi bazy danych podjęła grupa informatyków pod kierunkiem R. F. Canaday'a w BELL LABORATORIES, ogłaszając wyniki swoich prac w roku 1974 [1]. Nazwali oni taką konfigurację — w nawiązaniu do zastosowanej w teleprzetwarzaniu — „back-end configuration". Stąd wykorzystany system zarządzania bazą danych (SZBD) nazwano „back-end DBMS", a komputer realizujący funkcje SZBD — „back-end machine". Z braku ścisłego polskiego odpowiednika terminu „back-end" proponuję dla maszyny obsługującej bazę danych nazwę k o m p u t e r p o ś r e d n i c z ą c y bądź p o m o c n i c z y. Ponieważ jest on funkcjonalnie podporządkowany komputerowi centralnemu, całą konfigurację można określić jako rozwiązanie typu „master-slave" (MS). A zatem przez określenie „baza danych obsługiwana przez komputer pośredniczący (pomocniczy) będziemy rozumieli bazę danych funkcjonującą w takiej konfiguracji, w której funkcje SZBD zostały przeniesione z komputera centralnego na wyspecjalizowany komputer, służący wyłącznie do obsługi bazy danych i komunikacji z komputerem centralnym.

Jest oczywiste, że wprowadzenie dodatkowego komputera wymaga nowej organizacji oprogramowania. I tak oba komputery muszą być wyposażone w system komunikacyjny i oprogramowanie pośredniczące, natomiast komputer obsługujący bazę danych — oprócz SZBD — dodatkowo w tzw. zadania bazy danych, omówione w dalszej części artykułu.

System komunikacyjny służy do transmisji komend, danych oraz informacji o danych (metainformacji). Ponadto system ten kontroluje połączenia fizyczne, wykrywa i koryguje błędy transmisji, synchronizuje wykonanie zadań bazy danych oraz zarządza danymi w buforze komputera pośredniczącego.

Zasadniczą cechą systemu komunikacyjnego — poza niezawodnością — powinna być szybkość jego działania. W przypadku gdy szybkość ta będzie niedostateczna, czasy dostępu do bazy będą długie, zbliżone do osiąganych w warunkach konfiguracji jednomaszynowej. System komunikacyjny może więc stać się elementem ograniczającym sprawność konfiguracji MS. W pracach Canaday'a [1] i Maryansky'ego [3] pojawia się wielkość 1 Mb/s jako minimalna szybkość działania urządzenia komunikacyjnego otrzymana w wyniku badań symulacyjnych.

Drugim nowym elementem w konfiguracji MS są programy pośredniczące komputera centralnego i pomocniczego, które umożliwiają wymianę informacji między maszynami poprzez system komunikacyjny.

Działanie programu pośredniczącego komputera centralnego (PPC) rozpoczyna się w momencie, gdy program użytkowy zażąda dostępu do bazy danych. Wówczas PPC sprawdza, czy dany użytkownik ma prawo dostępu do danych, jeśli tak, to formułuje odpowiednie żądanie dla systemu komunikacyjnego i przesyła je do programu pośredniczącego komputera pomocniczego (PPP). Jako rezultat operacji wykonywanych na bazie danych, PPC otrzymuje dane i metainformacje, które przekształca do postaci zrozumiałej dla programu użytkowego i umieszcza w obszarze roboczym danego użytkownika. PPP natomiast odbiera komendy dostępu do danych z systemu komunikacyjnego i przekształca je na postać zadań bazy danych, zrozumiałą dla SZBD. Zadania bazy danych pełnią funkcję analogiczną do funkcji programów użytkowych w komputerze centralnym. Zawierają one bezpośrednie komendy dostępu do danych oraz obszar roboczy, do którego SZBD przesyła otrzymany wynik działania na bazie danych. PPP odbiera dane i metainformacje bądź komunikat o błędach z obszaru roboczego zadań bazy danych, formułuje je w sposób zrozumiały dla systemu komunikacyjnego i przesyła do PPC. Gdy system operacyjny komputera pośredniczącego jest wieloprogramowy i liczba zadań bazy danych zmienna, wówczas PPP musi mieć dodatkowo dostęp do mechanizmu alokacji i dealokacji zadań.

EFEKTYWNOŚĆ DZIAŁANIA SYSTEMU

Ocenie efektywności działania systemu należy poświęcić nieco więcej uwagi, ponieważ jest to jeden z czynników decydujących o celowości stosowania konfiguracji MS.

Zastosowanie komputera pośredniczącego jest wówczas zasadne, gdy dostęp do danych staje się wąskim gardłem systemu komputerowego (trwa zbyt długo), powodując przestoje jednostki centralnej komputera centralnego. Wiadomo jednak, że realizacja pojedynczej komendy dostępu do danych wymaga o wiele mniej czasu w konfiguracji jednomaszynowej niż w konfiguracji MS, gdzie żądanie dostępu do danych musi przejść przez dodatkowe moduły PPC, PPP i system komunikacyjny. Staje się więc jasne, jak istotne znaczenie — z punktu widzenia efektywności działania konfiguracji MS — mają wymienione elementy oprogramowania pośredniczącego oraz szybkość i poprawność działania systemu komunikacyjnego. Gdy system operacyjny jest jednoprogramowy (tzn. w pamięci operacyjnej komputera pośredniczącego może rezydować tylko jedno zadanie bazy danych), wówczas sprawność systemu komunikacyjnego i niezawodność oprogramowania pośredniczącego są jedynymi czynnikami decydującymi o efektywności działania całego systemu.

Zastosowanie komputera pomocniczego będzie sensowne jedynie wówczas, gdy czas przestoju wynikły z przeciążenia komputera centralnego będzie dłuższy niż czas potrzebny na realizację żądania dostępu do danych w konfiguracji MS. Dopiero gdy system operacyjny komputera pośredniczącego będzie wieloprogramowy, częstość odwołań programów użytkowych stanie się istotna, a dzięki wieloprogramowości systemów operacyjnych obu maszyn możliwe będzie równoległe wykonywanie programów użytkowych i operacji na bazie danych. Fakt, że realizacja pojedynczej komendy dostępu do danych bazy obsługiwanej przez osobny komputer trwa o wiele dłużej niż w konfiguracji jednomaszynowej, nie ma już w tych warunkach znaczenia, a sprawność systemu komunikacyjnego — nadal oczywiście istotna — nie jest jedynym czynnikiem decydującym o efektywności działania konfiguracji MS.

A zatem zastosowanie dodatkowej maszyny do obsługi bazy danych przyniesie oczekiwane rezultaty w postaci eliminacji przestojów procesora centralnego oraz niezależnego działania obu komputerów i pamięci zewnętrznych, gdy spełnione zostaną następujące warunki:

- system operacyjny komputera pośredniczącego będzie wieloprogramowy
- system komunikacyjny zapewni odpowiednio szybką i poprawną wymianę informacji pomiędzy obydwoma komputerami
- programy użytkowe będą wymagać częstego dostępu do bazy danych.

PRÓBY WDROŻENIOWE I ANALIZA PROTOTYPÓW

U progu lat osiemdziesiątych osiągnięcia w dziedzinie omawianych rozwiązań eksploatacji baz danych nie przedstawiają się imponująco. Podjęto wprawdzie kilka prób wdrożenia systemu komputerowego w konfiguracji MS, ale nic nie wskazuje na to, aby w najbliższej przyszłości komputery wyspecjalizowane w obsłudze baz danych były szeroko dostępne na światowym rynku informatycznym.

Najlepiej znane są cztery klasyczne już prototypy systemów komputerowych działających w konfiguracji MS: XDMS, IDMS, KSU i MADMAN.

Tabela sporządzona wyłącznie na podstawie pracy [3] najprawdopodobniej nie wyczerpuje listy obecnie działających systemów komputerowych w konfiguracji MS, niemniej jednak brak aktualnych danych uniemożliwia jej rozszerzenie.

Na szczególną uwagę zasługuje system MADMAN, w którym procesory komputera centralnego i pośredniczącego są kompatybilne i działają w oparciu o wspólną pamięć. Komputer centralny nadal inicjuje operacje wejścia-wyjścia, wykonuje operacje na dyskach oraz przesyła dane do i ze swojej pamięci operacyjnej, która jest jednocześnie pamięcią komputera pośredniczącego. Dzięki połączeniom zwanym „unibus linkages" maszyna pośrednicząca ma możliwość jednoczesnego dostępu do buforów zawierających komendy dostępu do danych oraz informacji przekazywanych i otrzymywanych z bazy danych. Komputer pośredniczący LSI-11 jest wieloprogramowy, a zatem może wykonywać operacje w jednostce centralnej i współuczestniczyć w wykonywaniu operacji wejścia-wyjścia. Taka konfiguracja umożliwia bardzo szybką wymianę informacji między procesorami i znacznie upraszcza oprogramowanie niezbędne do komunikacji między obydwoma maszynami. Efektywność działania systemu MADMAN zależy głównie od wielkości pamięci operacyjnej i sprawności działania obu procesorów. W systemie MADMAN zastosowano system operacyjny IAS.

Zupełnie inne rozwiązanie sprzętowe zaproponowali autorzy systemu KSU, gdzie komputerem centralnym jest minikomputer INTERDATA 8/32 — współpracujący bądź z minikomputerem INTERDATA 7/82, bądź z dużym komputerem ITEL AS5. Dokładne badania obu konfiguracji systemu KSU wykazały, że konfiguracja, w której maszyną pośredniczącą jest ITEL AS5, jest o 41% wydajniejsza niż konfiguracja z minikomputerem INTERDATA 7/32.

Oprogramowanie pierwotnej wersji systemu KSU było napisane w różnych językach: SZBD TOTAL — w asemblerze, programy użytkowe i pośredniczące — w COBOLU, system komunikacyjny dla maszyny INTERDATA 7/32 — w CONCURRENT PASCAL, a dla ITEL AS5 — w asemblerze. Tak zróżnicowane oprogramowanie znacznie utrudniało rozwój poszczególnych modułów i wydłużało czas dostępu do danych. Dlatego też w drugiej wersji systemu KSU oprogramowanie wykonano tylko z użyciem dwóch języków: CONCURRENT PASCAL i SEQUENTIAL PASCAL, a jako komputer pośredniczący wykorzystano jedynie minikomputer INTERDATA 7/32.

W efekcie czas odpowiedzi systemu na żądanie dostępu do bazy danych w porównaniu do pierwszej wersji skrócono od półtora do trzech razy.

Badania nad wymienionymi prototypami konfiguracji MS potwierdzają słuszność przedstawionych założeń dotyczących wieloprogramowości systemów operacyjnych obu maszyn, szybkiego systemu komunikacyjnego i częstych odwołań do bazy danych z programów użytkowych. Wyniki testów eksploatacyjnych systemu XDMS wykazały, że dopiero realizacja złożonych komend dostępu [1]) do danych w konfiguracji MS istotnie odciąża komputer centralny i przynosi oczekiwany efekt czasowy, natomiast realizacja komend prostych jest szybsza w konfiguracji jednomaszynowej. Z kolei pracę systemu IDMS badana w warunkach jedno- i wieloprogramowości systemu operacyjnego. Okazało się, że czas wykonania jednego programu użytkowego w jednoprogramowym trybie pracy jest równy czasowi wykonania trzech takich programów w trybie wieloprogramowym.

Dokładne statystyki na temat szybkości działania konfiguracji MS, w zależności od zastosowania jedno-, trój- i pięcioprogramowego systemu operacyjnego maszyny obsługującej bazę danych, można znaleźć w pracy [4].

Szczegółowa analiza działania prototypów konfiguracji MS wykracza jednak poza zakres niniejszego opracowania. Dokładne informacje na ten temat można znaleźć w podanych pozycjach literatury.

---

[1]) Komenda dostępu do danych była oceniana jako złożona zależnie od liczby operacji, jakie trzeba było wykonywać na urządzeniach pamięci zewnętrznej oraz od tego, jaki czas pracy procesora zajmowała realizacja komendy dostępu

Charakterystyka przykładowych systemów komputerowych w konfiguracji MS

Nazwa systemu	Wykonawca	Komputer centralny	Komputer pośredniczący	SZBD	Szybkość transmisji danych	Zastosowanie
XDMS	BELL LABORATORIES	UNIVAC 1108	META-4	DMS 1100	2000 b/s	egzemplarz eksperymentalny
IDMS IDMS11	Kansas State University CULLINANE CORPORATION	IBM 370/158	PDP/11	IDMS	4800 b/s	działalność ekonomiczna Armii USA
KSU	Kansas State University	INTERDATA 8/32	INTERDATA 7/32 ITEL AS5	TOTAL	linia telefoniczna	działalność finansowo-księgowa dla wewnętrznych potrzeb Armii USA
MADMAN	GENERAL ELECTRIC	PDP 11	LSI-11	MADMAN	wspólna pamięć	działalność finansowo-księgowa w przedsiębiorstwie produkcyjnym

Ogólna ocena prób wdrożenia systemów komputerowych wykorzystujących dodatkowy komputer do obsługi baz danych nie jest jednoznaczna. Wyniki badań eksploatacyjnych nie potwierdziły teoretycznych możliwości, jakie daje zastosowanie komputera pośredniczącego. Główną przyczyną takiego stanu rzeczy były chyba zbyt wolne połączenia komunikacyjne między komputerem centralnym a pośredniczącym (4,8 Kb/s w IDMS, 2 Kb/s w XDMS, 1,2 lub 2,4 Kb/s w KSU), natomiast dotychczasowe testy systemu MADMAN nie upoważniają jeszcze do stwierdzenia, że częściowe zastąpienie systemu komunikacyjnego wspólną dla obu procesorów pamięcią jest lepszym rozwiązaniem. Można mieć nadzieję, że zwiększenie szybkości przepływu informacji do proponowanej wielkości 1 Mb/s przyniesie oczekiwane efekty. Nie można jednak w oparciu o badania symulacyjne stwierdzić, o ile „lepsze" będzie wtedy działanie konfiguracji MS.

Dotychczasowe doświadczenia związane z eksploatacją prototypów konfiguracji MS nie są więc wystarczające — przynajmniej w przekonaniu producentów — do wyjścia na rynek z produkcją seryjną komputerów wyspecjalizowanych w obsłudze baz danych. Z drugiej jednak strony dużym osiągnięciem autorów i realizatorów omawianych prototypów jest zebranie argumentów świadczących o sensowności dalszych badań nad taką konfiguracją.

Z dwóch co najmniej powodów warto szukać lepszych rozwiązań konstrukcyjnych konfiguracji MS.

Powód pierwszy jest czysto ekonomiczny: wprowadzenie komputera wspomagającego komputer centralny jest bardziej opłacalne niż rozbudowa istniejącego lub zakup nowego komputera o większej mocy obliczeniowej. Obecnie różnica w cenie między minikomputerami a komputerami o dużej mocy obliczeniowej jest na tyle duża, że opłaca się ponieść koszty dodatkowego oprogramowania sprzęgającego oba komputery. Mimo stałej tendencji obniżki ocen sprzętu, ich proporcje między dużymi a małymi komputerami pozostaną niezmienione, a nawet przechylają się na korzyść sprzętu mini- i mikrokomputerowego, który tanieje szybciej.

Drugim powodem, dla którego warto jest kontynuować badania nad konfiguracją MS jest znaczna oszczędność pamięci operacyjnej komputera centralnego. Oprogramowanie związane z obsługą bazy danych (schemat, podschematy, bufory i SZBD) zajmują oczywiście o wiele więcej miejsca w pamięci operacyjnej niż PPC (5K w systemie KSU) oraz część oprogramowania systemu komunikacyjnego przypadająca na komputer centralny.

Dotychczasowe doświadczenia eksploatacji prototypów konfiguracji MS nie pozwalają niestety powiedzieć czegokolwiek na temat zwiększenia wrażliwości systemu na awarie sprzętu i oprogramowania oraz na temat wzmocnienia systemu ochrony danych.

* * *

Na koncepcję bazy danych obsługiwanej przez osobny komputer można również patrzeć całkiem inaczej, a mianowicie — jak na pierwszy etap rozproszenia inteligencji systemu komputerowego i tworzenia konstrukcji specjalizowanych systemów pamięci, tzn. takich systemów, gdzie dostęp do danych jest realizowany sprzętowo. Punktem docelowym tego kierunku rozważań teoretycznych jest tzw. „database machine", czyli baza danych, która sama siebie obsługuje i kontroluje. Obecnie jednak rozważania na ten temat są na tyle odległe od rzeczywistości, że przypominają raczej agitację, a funkcjonujące prototypy nie pozwalają jeszcze na wysnucie konkretnych wniosków.

Warto natomiast wspomnieć o ciekawym pomyśle zastosowania konfiguracji MS w dziedzinie przetwarzania rozproszonego. Możliwe jest bowiem rozwiązanie, w którym jedna maszyna pośrednicząca współpracuje z kilkoma, rozproszonymi geograficznie komputerami centralnymi. Jeżeli uda się pokonać bariery technologiczne (głównie komunikacyjne), wówczas będzie można osiągnąć korzyści płynące zarówno z centralnego zarządzania danymi, jak i z rozproszonego przetwarzania.

LITERATURA

[1] Canaday R. H., Harrison R. D., Ivie E. L., Rydel J. L., Wehr L. A.: A Backend Computer for Data Base Management. CACM 1974, 17 No. 10

[2] Date C. J.: Wprowadzenie do baz danych. PWN, 1981

[3] Maryanski F. J.: Backend Database Systems. ACM Comp. Surv. 1980, vol. 12, No. 1

[4] Maryanski F. J., Fisher P. S., Wallentine V. E.: An Evaluation of a Conversion to a Backend Database Management System. Proc. ACM Ann. Conf. Oct. 1976

[5] Ross R.: Data Base Systems: Design, Implementation and Management. AMACOM 1978.

Zasady prenumeraty

Zamówienia i przedpłaty na prenumeratę INFORMATYKI przyjmuje Zakład Kolportażu Wydawnictwa NOT SIGMA. Adres pocztowy: Wydawnictwo NOT SIGMA — Zakład Kolportażu, 00-950 Warszawa, skr. poczt. 1004. Konto bankowe: III O/M NBP Warszawa nr 1036-7490-139-11.

PRENUMERATORZY ZBIOROWI — jednostki gospodarki uspołecznionej, instytucje i organizacje — składają zamówienia w formie wpłaty-przelewu na specjalnym blankiecie opracowanym przez Wydawnictwo. Część tego blankietu zawiera listę tytułów czasopism kolportowanych przez Wydawnictwo.

W przypadku większej liczby odbiorców:

— od 2 do 5 — należy dokonać wpłaty oddzielnie dla każdego odbiorcy

— od 6 odbiorców — dokonać jednej wpłaty, z powołaniem się na znak kancelaryjny pisma przesłanego do Zakładu Kolportażu, a zawierającego wykaz adresów poszczególnych odbiorców i zamawianych dla nich tytułów czasopism oraz numer konta bankowego, z którego dokonany został przelew. Na wpłacie-zamówieniu należy podać łączne liczby egzemplarzy poszczególnych tytułów czasopism.

WPŁATY-ZAMÓWIENIA przyjmowane są w terminach:

- **do 15 listopada — na I kwartał, I półrocze i cały rok następny**
- **do 28 lutego — na II, III i IV kwartał**
- **do 31 maja — na II półrocze i IV kwartał**
- **do 31 sierpnia — na IV kwartał**

PRENUMERATA STAŁA-WIELOLETNIA dotyczy tylko prenumeratorów zbiorowych. Zamawiający będzie otrzymywał z Wydawnictwa potwierdzenie kontynuacji prenumeraty wraz z wezwaniem do zapłaty. Zmiany w prenumeracie należy zgłaszać pisemnie w terminach obowiązujących dla składania zamówień-wpłat.

PRENUMERATORZY INDYWIDUALNI — osoby fizyczne — zamawiają prenumeratę dokonując wpłaty na blankiecie opracowanym przez Wydawnictwo NOT SIGMA (dostępnym w urzędach pocztowych) lub na blankiecie przekazu NBP — w terminach j.w. Na odwrocie środkowego odcinka przekazu, przeznaczonego dla adresata-posiadacza rachunku (Wydawnictwa), należy podać tytuł zamawianego czasopisma, okres prenumeraty i liczbę egzemplarzy.

Ceny INFORMATYKI

Cena egz.	Prenumerata w złotych					
	Normalna			Ulgowa		
100,—	kwart.	półrocz.	roczna	kwart.	półrocz.	roczna
	300,—	600,—	1200,—	105,—	210,—	420,—

(dokończenie na str. 21)

ZAKŁAD TECHNIKI BIUROWEJ PREDOM–ORG

50-950 Wrocław, ul. Chełmońskiego 10, tel. centr. 48-42-93, teleks 0715153

rozpowszechnia w kraju

MIKROKOMPUTERY serii ELWRO 500

Organizacja PREDOM-ORG, posiadająca na terenie kraju 8 zakładów i około 100 oddziałów oraz punktów usługowych, prowadzi:

— dystrybucję
— instalację
— serwis gwarancyjny i pogwarancyjny
— centralną bibliotekę oprogramowania użytkowego
— dostawę części zamiennych i materiałów eksploatacyjnych (poza papierem tabulogramowym)

W 1984 r. PREDOM–ORG oferuje mikrokomputer ELWRO 523 o następujących cechach:

— pamięć operacyjna 48 KB
— budowa modułowa: klawiatura, pamięć zewnętrzna na dyskach elastycznych 8", monitor ekranowy, drukarka z przystawką do kart kontowych
— system operacyjny EMOS kompatybilny z CP/M wersja 2.2 wraz z oprogramowaniem podstawowym: program sterujący, ładujący, interpreter języka ZIM i BASIC
— możliwość zastosowania w pracach ewidencyjno-księgowych do fakturowania, rozliczeń finansowych i bankowych, technicznego przygotowania produkcji, rejestrowania i przetwarzania danych.

Zamówienia przyjmuje **Dział Handlowy ZTB PREDOM-ORG 50-950 Wrocław, ul. Chełmońskiego 10, tel. 48-63-51 (główny dystrybutor)**, który udziela również informacji w sprawie warunków sprzedaży i dostaw.

Szczegółowe informacje w sprawie oprogramowania użytkowego można uzyskać w **Ośrodku Projektowania Przetwarzania Danych PREDOM-ORG, 50-411 Wrocław, Al. Słowackiego 35, pok. 200, tel. 341-31.**

Dystrybucję mikrokomputerów serii ELWRO 500 na terenie całego kraju prowadzą:

PTB, 01-244 Warszawa, ul. Bema 57a, tel. 32-12-31
ZTB, 20-704 Lublin, ul. Wojciechowska 3a, tel. 500-17 do 19
ZTB, 40-064 Katowice, ul. Kopernika 26-28, tel. 51-84-91 i 51-80-11
ZTB, 80-341 Gdańsk, ul. Bałtycka 5, tel. 93-09-15
ZTB, 60-750 Poznań, ul. Wyspiańskiego 15a, tel. 513-73
ZTB, 30-114 Kraków, ul. Kościuszki 43, tel. 22-10-96
ZTB, 93-176 Łódź, ul. Suwalska 16, tel. 84-27-30 do 32

Zamówienia będą realizowane w kolejności zgłoszeń. PREDOM-ORG gwarantuje dobrą jakość sprzętu, usług technicznych i oprogramowania.

ZGŁOŚ PROBLEM
OTRZYMASZ ROZWIĄZANIE WRAZ ZE SPRZĘTEM
I ZAPEWNIENIEM CAŁOŚCIOWEJ OBSŁUGI

EO/1099/K/84

CEZARY ZIELIŃSKI
Instytut Automatyki
Politechnika Warszawska

Roboty (4)

Złożone języki programowania

Złożone języki programowania robotów wykorzystują model środowiska, w odróżnieniu od języków prostych, które korzystają z modelu manipulatora bądź modeli części łańcucha kinematycznego stanowiącego ramię. Zarówno w językach zorientowanych na przemieszczanie końcówki manipulatora, jak i w językach zorientowanych na przemieszczanie obiektów — operuje się pojęciem układu współrzędnych. Istnieje jednak poważna różnica pomiędzy obydwoma sposobami jego użycia.

W pierwszej grupie układ współrzędnych reprezentuje pewne charakterystyczne położenia w przestrzeni kartezjańskiej, odnoszące się do narzędzia zamontowanego na ramieniu. Drugorzędną sprawą jest natomiast fakt, że układy współrzędnych są związane z obiektami znajdującymi się w środowisku. W tym przypadku robot nie ma żadnej informacji o relacjach zachodzących pomiędzy poszczególnymi układami. Przykładowo — o ile pewnemu przedmiotowi przyporządkujemy dwa układy współrzędnych: jeden związany z jego podstawą, a drugi z miejscem chwytu, to robot ujmując obiekt za chwytnię, a następnie przemieszczając go, nie będzie miał „świadomości", że układ związany z podstawą także doznał przesunięcia. Dlatego uważa się, że następuje tu jedynie przesuwanie ramienia pomiędzy różnymi położeniami, a nie przemieszczanie obiektów.

Systemy ukierunkowane na przemieszczanie obiektów mają tak skonstruowany model otoczenia, że „ruch" pewnego układu współrzędnych powoduje jednoczesne uaktualnienie pozycji pozostałych układów związanych z przemieszczanym obiektem. Co więcej — gdy na poruszanym przedmiocie znajduje się inny, to jego układy współrzędnych również ulegną modyfikacji. Jest to zupełnie inne podejście do zagadnienia programowania robotów.

Języki ukierunkowane na zadanie tym różnią się od poprzednio opisanych, że oprócz modelu samego środowiska zapewniają również możliwość budowania planu działania. Języki te muszą zawierać informacje o tym, jak dekomponować zadanie na czynności podstawowe oraz jak wybrać optymalny (ewentualnie suboptymalny) sposób osiągnięcia celu. Dlatego też są one językami sztucznej inteligencji albo bezpośrednio się z nich wywodzą.

*

Jednym z najstarszych oraz najbardziej znanych języków wysokiego poziomu przeznaczonych do programowania robotów jest AL (Assembly Language — język montażowy). Jest on ukierunkowany na przemieszczanie obiektów. Jego podstawowe założenia zostały opracowane w Stanford Artificial Intelligence Laboratory w 1974 roku. Przy jego opracowaniu korzystano z doświadczeń uzyskanych w trakcie implementacji języka WAVE. W miarę upływu lat AL był modyfikowany, tak by z jednej strony stał się narzędziem maksymalnie elastycznym, a z drugiej — był łatwy w nauczaniu. Ponieważ istnieją już na rynku, aczkolwiek zawężone, wersje użytkowe tego języka, to wydaje się, iż zasługuje on na szersze omówienie.

Cały system sterowania został tak pomyślany, że można za jego pomocą sterować wieloma ramionami robotów oraz urządzeniami współpracującymi. Poszczególne funkcje zostały rozdzielone pomiędzy dwa komputery: PDP-KL 10 oraz PDP 11/45. Pierwszy z nich zajmuje się redagowaniem, kompilacją oraz ładowaniem programów napisanych w języku AL, natomiast drugi odpowiedzialny jest za wykonanie kodu wynikowego oraz bezpośrednie sterowanie ramieniem. Kompilator dokonuje zarówno analizy leksykalnej, syntaktycznej oraz generacji kodu wynikowego, jak i symulacji programu, dzięki czemu może wykrywać błędy semantyczne. Przykładowo — wykrycie przyszłego zderzenia ramienia z podłożem nie stanowi problemu i jest sygnalizowane jako błąd semantyczny.

AL został tak zaprojektowany, aby jego typy danych umożliwiały łatwe zaprogramowanie ruchu ramienia manipulatora, operowanie trójwymiarowymi obiektami oraz odzwierciedlenie relacji zachodzących między przedmiotami. Zrezygnowano natomiast z typów danych tradycyjnie stosowanych w komputerowych językach wysokiego poziomu.

Podstawowym typem jest **skalar.** W maszynie jest on reprezentowany jako liczba zmiennoprzecinkowa. Zmienna skalarna prócz swej wartości, będącej liczbą rzeczywistą, może również posiadać miano, czyli jednostkę fizyczną. W związku z tym możemy mieć do czynienia ze skalarami siły, odległości, czasu lub jakimikolwiek innymi. W trakcie kompilacji wyrażeń sprawdzana jest zgodność jednostek fizycznych i w przypadku jej braku — następuje sygnalizacja błędu. Skalary mianowane służą do reprezentacji odległości, siły lub szybkości, natomiast niemianowane mogą być wykorzystywane jako liczniki pętli. Jak widać, całkowicie zrezygnowano tu z typowego podziału na liczby INTEGER oraz REAL.

Na skalarach można przeprowadzać operacje dodawania, odejmowania, mnożenia, dzielenia, potęgowania. Ponadto mogą one stanowić argumenty wielu funkcji zdefiniowanych w systemie.

Drugim typem danych jest **wektor.** Służy on do przedstawiania wielkości wektorowych, a reprezentowany jest jako uporządkowana trójka skalarów. Na zmiennych tego rodzaju można wykonywać operacje dodawania, odejmowania oraz realizować trzy typy mnożenia. Może to być więc mnożenie przez skalar, iloczyn skalarny oraz iloczyn wektorowy. W toku translacji operacji matematycznych badana jest poprawność mian w wyrażeniu.

Trzecim typem danych jest **obrót** lub **orientacja przestrzenna** (ang. rotation). Aby określić przekształcenie polegające na obrocie wektora lub układu współrzędnych w przestrzeni, podajemy wektor wyznaczający oś obrotu oraz skalar determinujący kąt rotacji. Język dopuszcza składanie kilku uprzednio zdefiniowanych obrotów w jeden bardziej złożony. Oprócz tego istnieją funkcje dokonujące ekstrakcji osi oraz kąta obrotu. W związku z tym, że rotacja jest przekształceniem, nie ma ona miana.

Obiekty znajdujące się w środowisku mają zarówno położenie, jak i orientację. W związku z tym wprowadzono do AL typ danych — nazywany **trójścianem** albo **układem współrzędnych,** a także **pozycją** (ang. coordinate frame) — który umożliwia specyfikację cech przestrzennych przedmiotów. Aby pozycji nadać wartość, należy podać wektor, który określa położenie początku nowo tworzonego układu współrzędnych, oraz obrót, który determinuje orientację poszczególnych jego osi. Trójściany można przesuwać w przestrzeni dodając lub odejmując do i od nich wektory. Można także zmieniać ich orientację — mnożąc je przez zmienne typu obrót. Ponadto można wyznaczać współrzędne wektora wyrażonego w jednym układzie współrzędnych przez współrzędne innego układu.

Ostatni z prostych typów danych jest syntezą wektora i obrotu. Nazwano go **transformacją** (ang. transformation).

Mnożąc lewostronnie wektor lub trójścian przez transformację, powodujemy ich przesunięcie oraz odpowiednią zmianę orientacji. Transformacje, podobnie jak pozycje, składają się z wektora oraz obrotu. Identyczna jest też ich reprezentacja maszynowa — istnieją więc tendencje, aby nie wprowadzać rozróżnienia między tymi typami danych.

Jedynym złożonym typem danych jest **tablica**. Jej elementy są typu prostego, a więc istnieją tablice skalarów, wektorów, orientacji, pozycji albo transformacji. W języku nie wprowadzono żadnych ograniczeń na liczbę indeksów tablicy.

Ten krótki przegląd typów danych języka AL zakończymy przykładem ilustrującym składnię deklaracji typów (w nawiasach klamrowych znajdują się komentarze).

```
SCALAR s1; {s1 jest skalarem niemianowanym}
DISTANCE SCALAR ds; {mianem ds jest jednostka długości}
ANGLE SCALAR fi; {fi będzie wykorzystywane jako kąt}
DISTANCE VECTOR dv;
ROT r1: {zadeklarowano obrót r1}
FRAME f1, f2; {zadeklarowano trójścian f1 oraz f2}
TRANS tt; {zadeklarowano transformację}
s1 ← 5; {s1 nadano wartość 5}
ds ← 2*cm; {ds nadano wartość 2 cm}
fi ← 90*deg; {fi nadano wartość 90°}
dv ← VECTOR (1, 2, 3)*cm; {utworzono wektor [1, 2, 3] cm}
r1 ← ROT(dv, fi); {r1 stanowi obrót wokół osi wyznaczonej
     przez wektor [1, 2, 3] o 90°}
f1 ← FRAME(ROT (zhat,90*deg), 2*xhat*cm);
     {f1 jest trójścianem obróconym o 90° wokół osi z układu bazowego oraz ma swój początek przesunięty o 2
     cm wzdłuż osi x tegoż układu}
tt ← station → f1; {tt jest przekształceniem transformującym układ bazowy (station) na pozycję f1}
f2 ← tt*f1; {utworzono układ współrzędnych f2 transformując f1 przekształceniem tt}
```

Oczywiście, w każdej z powyższych deklaracji można by użyć bardziej złożonych wyrażeń — korzystając z operacji zdefiniowanych na poszczególnych typach danych. Nie uczyniono tak, aby nie komplikować przykładu.

Struktura programów w ALU jest blokowa. Nie różni się ona niczym od struktury programów napisanych w ALGOLU. Wszystkie zasady związane z regułami tworzenia i widoczności zmiennych zostały stamtąd zapożyczone.

Instrukcje sterowania wykonaniem programu są także znane z innych języków programowania. Są nimi konstrukcje: IF THEN ELSE, FOR, WHILE, UNTIL oraz CASE. Istnieje również możliwość korzystania z procedur oraz procesów. Synchronizację między procesami zapewnia mechanizm semaforowy (EVENT, SIGNAL, WAIT).

Podstawowym rozkazem ruchu ramienia robota jest instrukcja MOVE. Jej składnia jest następująca:

MOVE <przemieszczalny układ współrzędnych> TO <pozycja końcowa> <warunki modyfikujące>;

Przez **<przemieszczalny układ współrzędnych>** rozumie się albo trójścian związany z końcówką manipulatora, albo pozycję związaną z obiektem uchwyconym uprzednio przez robota. **<Pozycja końcowa>** określa trójścian, z którym mają się pokryć płaszczyzny rozpięte na osiach przemieszczalnego układu współrzędnych. Natomiast **<warunki modyfikujące>** zawierają dodatkową informację o sposobie ruchu.

Projektanci systemu przewidzieli wiele modyfikacji ruchu podstawowego. Mogą to być: „pozycja odejścia" od aktualnie zajmowanego przez końcówkę położenia (WITH DEPARTURE) lub „pozycja pośrednia" przy dochodzeniu do celu (WITH APPROACH). Możliwe jest ponadto wyszczególnienie wielu układów współrzędnych, przez które ma zostać „przeniesiony" trójścian związany z narzędziem (VIA). Stosując warunek WITH DURATION określa się czas, w którym ruch ma być wykonany. Można także ograniczyć prędkość wykonania ruchu, modyfikując rozkaz ruchu warunkiem WITH SPEEDFACTOR. W przypadku wkładania lub wyjmowania ciasno spasowanych elementów z otworów, pomocne jest wprawienie nadgarstka w drgania. Umożliwia to warunek WITH WOBBLE. Ponieważ nadgarstek robota zastosowanego przy implementacji systemu zaopatrzony jest w czujniki siły oraz momentów siły, możliwy stał się ruch z jednoczesnym wywieraniem odpowiedniego nacisku (WITH FORCE) lub momentu siły (WITH TORQUE).

Manipulator stosowany w Stanford Artificial Intelligence Laboratory zaopatrzony jest w chwytak, którego palce wyposażone są w czujniki dotyku. Odległość rozwarcia mechanizmu chwytającego może być ustawiona dowolnie. W związku z tym realizatorzy ALA nie mieli kłopotów z implementacją rozkazów OPEN, CLOSE oraz CENTER, zaczerpniętych z języka WAVE.

W systemie przewidziano również sterowanie urządzeniami współpracującymi z robotem. Do tego celu służy dwuargumentowa instrukcja OPERATE. Pierwszy z argumentów określa nazwę urządzenia, którego dotyczy ten rozkaz, drugi natomiast — czynności, jaką ma ten automat wykonać.

Tym, co z języka AL czyni język ukierunkowany na przemieszczanie obiektów, jest możliwość modelowania środowiska, w którym robot pracuje. Model „świata" (WORLD MODEL), w którym działa manipulator, tworzony jest przez określenie obiektów, ich pozycji oraz relacji zachodzących między nimi. Przez obiekt rozumiemy tu autonomiczny przedmiot, jego charakterystyczny fragment lub konstrukcję składającą się z wielu podzespołów. Modelowaniem relacji między obiektami zajmują się instrukcje AFFIX oraz UNFIX.

Ogólna postać pierwszej z nich jest następująca:

AFFIX <f1> TO <f2> BY <t> AT <wyrażenie> <modyfikacja>

<f1> oraz <f2> są układami współrzędnych związanych z obiektami, natomiast <t> oraz **<wyrażenie>** — transformacjami. Rozkaz ten powoduje, że trójścian <f1> zostanie przytwierdzony do <f2> w pozycji opisanej przez **<wyrażenie>**. Zmiennej <t> przypisywana jest wartość przekształcenia <f2> na <f1>. Instrukcja ta ma dwie odmiany określone **<modyfikacją>**. Przytwierdzenie może być sztywne (RIGIDLY) oraz jednostronne (NON_RIGIDLY). Z pierwszym przypadkiem mamy do czynienia, gdy dwa obiekty zostaną sklejone, skręcone lub gdy <f1> i <f2> są dwoma różnymi fragmentami tego samego obiektu. Dla systemu instrukcja ta stanowi informację, że każde przemieszczenie <f1> spowoduje ruch <f2> i odwrotnie.

Z drugim przypadkiem spotykamy się kładąc jeden przedmiot na drugim. Wtedy przesuwając <f2> zmieniamy także położenie <f1>, ale przemieszczając <f1> nie poruszamy <f2>. Ponieważ do trójścianów odwołujemy się przez ich nazwy, zazwyczaj takie same jak nazwy obiektów, to faktycznie operujemy przedmiotami. Argumentem instrukcji MOVE może być dowolny przemieszczany, to znaczy uchwycony obiekt — trójścian. W związku z tym, programista zajmuje się manipulacją obiektami, natomiast system dokonuje odpowiednich pozycjonowań ramienia robota oraz śledzi zmiany zachodzące w środowisku, czyli położenia przesuwanych przedmiotów. Najlepiej będzie zobrazować to na przykładzie.

AFFIX klocek_a TO klocek_b BY t AT TRANS (nilrot, VECT (1,1,4)) NON_RIGIDLY;

Załóżmy, że przed wykonaniem tej instrukcji **klocek_a** został postawiony na **klocku_b**. Rozkaz ten poinformuje system, że **klocek_a** spoczął na **klocku_b** i każde przemieszczenie tego drugiego spowoduje ruch pierwszego, ale nie na odwrót.

MOVE klocek_b to fff WITH DURATION 10*s;

Jeżeli **klocek_b** był uchwycony, to rozkaz ten przemieści oba klocki tak, aby **klocek_b** zajął pozycję fff. Po otwarciu chwytaka można zażądać:

MOVE ramie TO klocek_a:

Koniec ramienia zostanie przemieszczony do pozycji umożliwiającej uchwycenie **klocka_a**. Gdyby teraz chcieć przemieścić jedynie **klocek_a**, to należy poinformować system o zerwaniu relacji jednostronnego związania obiektów. Dokonuje tego instrukcja;

UNFIX <f1> FROM <f2>;

Jak widać z powyższego prostego przykładu, programista zajmuje się przede wszystkim przemieszczaniem obiektów, a więc dokonywaniem czynności montażowych, natomiast w małym stopniu jest zajęty samym manipulatorem i jego ruchem.

*

Dalszym rozwinięciem języka AL jest POINTY. Programiści piszący programy dla robota w języku AL stwierdzili, że specyfikacja układów współrzędnych związanych z poszczególnymi obiektami jest zarówno uciążliwa, jak i podatna na błędy. Dlatego też realizatorzy POINTY wprowadzili nowy mechanizm, usuwający te mankamenty. Umożliwiono definiowanie trójścianów za pomocą manipulatora, a dokładniej — wskaźnika uchwyconego przez manipulator. Ponieważ niekiedy wiąże się układy współrzędnych z trudno dostępnymi fragmentami obiektów, na przykład z wnętrzami otworów — skonstruowano wskaźnik, który można wyginać (ang. bendy pointer). Wskaźnik ten jest tak wykonany, że można go dowolnie ukształtować; natomiast w trakcie pracy, przy stykaniu się z obiektami czy na skutek działania sił grawitacji — kształt jego nie jest tracony.

Po każdym wygięciu, wskaźnik musi zostać skalibrowany. Do tego służy instrukcja podstawienia:

FIDUCIAL ← <ramię>;

przy czym **FIDUCIAL** jest dowolnie wybranym stałym punktem uprzednio zdefiniowanym w systemie. Po wygięciu wskaźnika, ramię ustawia się w ten sposób, aby wskazywało ten punkt. Wykonanie rozkazu wzorcuje wskaźnik.

Po skalibrowaniu wskaźnika można przystąpić do definiowania poszczególnych układów współrzędnych oraz relacji zachodzących między nimi, a więc do określenia modelu środowiska. Podstawową instrukcją, pomocną w tym względzie, jest **FCONSTRUCT <pozycja>;** — umożliwiająca nadanie wartości trójścianowi. Wartość pozycji tworzona jest z trzech odczytów położenia wskaźnika, a więc i ramienia manipulatora. Wskazanie pierwsze określa początek nowo definiowanego układu współrzędnych, dwa następne wyznaczają jedną z osi tego układu oraz płaszczyznę między tą osią a następną. Programista sam określa te osie. Relacje zachodzące między poszczególnymi trójścianami określa się, tak jak w ALU, za pomocą instrukcji AFFIX oraz UNFIX.

W ten sposób budowanie modelu środowiska, a więc i pisanie programu, stało się interakcyjne. W trakcie konstruowania programu można zażądać wykonania pewnego jego fragmentu dla sprawdzenia poprawności. Ponieważ pozostałe rozszerzenia nie wprowadzają istotnych zmian w porównaniu z ALEM, nie będziemy się tu nimi zajmować.

*

Język AL doczekał się wielu naśladowców. Z istotniejszych można wymienić PAL (Purdue Arm Language lub Purdue Assembly Language) opracowany w Purdue University. Przy jego definiowaniu obnażono strukturę abstrakcyjnej reprezentacji poszczególnych typów danych, a więc to, co było starannie ukryte przed użytkownikiem ALA. Ponieważ każdy układ współrzędnych, każdy obrót czy transformacja mogą być sprowadzone poprzez przekształcenie jednorodne (ang. homogeneous transformation) do macierzy 4×4, to operacje w tym języku są zapisywane jako mnożenie lub odwracanie macierzy. Przykładowo:

OBIEKT = Z * T6 * E

Z reprezentuje pozycję podstawy manipulatora względem układu bazowego, **T6** — pozycję nadgarstka względem podstawy, a **E** — pozycję końca narzędzia względem nadgarstka. Ponieważ **Z** oraz **E** są stałe dla danego programu i narzędzia, to znając pozycję OBIEKTU można wyliczyć **T6**, a co za tym idzie — położenia poszczególnych fragmentów łańcucha kinematycznego, stanowiącego ramię robota. Dokonuje tego oczywiście, układ sterujący. Aby więc manipulator zajął pozycję umożliwiającą uchwycenie OBIEKTU, wystarczy podać proste równanie. System dopuszcza bardziej uwikłaną formę równań, ponieważ operacja mnożenia i odwracania macierzy jest dozwolona po obu stronach równości.

Język ten zawiera również instrukcje ruchu do pozycji (**MOV <pozycja>**) oraz rozkazy oddziałujące na narzędzie (w przypadku chwytaka — GRASP oraz RELEASE).

*

Językiem także częściowo wzorowanym na ALU jest PASRO (Pascal for Robots), opracowany w RFN. Stanowi on znacznie uproszczoną wersję ALA. Podstawowe uproszczenie polega na odrzuceniu instrukcji **AFFIX** oraz **UNFIX** a co za tym idzie — zrezygnowaniu z możliwości definiowania relacji między poszczególnymi obiektami (trójścianami). Dlatego też języka tego nie można zakwalifikować do grupy ukierunkowanej na przesunięcia obiektów.

PASRO został zrealizowany jako język „zanurzony w PASCALU". Wszystkie specjalizowane typy danych zostały więc zdefiniowane jako rekordy. Przykładowo:

```
TYPE vector = RECORD
                x, y, z: REAL
              END;
TYPE rotation = RECORD
                axis:vector;
                angle:REAL;
                matrix:rotmatrix
              END;
TYPE rotmatrix = RECORD
                   t, o, a:vector
                 END;
TYPE frame = RECORD
               rot:rotation;
               transl:vector
             END;
```

Natomiast instrukcje powodujące ruch ramienia są procedurami bibliotecznymi. Niektóre z nich to:

PMOVE(fin:frame); — powoduje ruch PTP do pozycji **fin.**
JMOVE(fin:frame); — wywołuje ruch do pozycji **fin** z liniową interpolacją we współrzędnych naturalnych robota
SMOVE(fin:frame); — przemieszcza ramię do **fin** z interpolacją liniową we współrzędnych kartezjańskich.

Poprzednie języki zostały tak zaprojektowane, aby mogły służyć pracom badawczym; natomiast PASRO został zrealizowany w celu czysto komercjalnym.

* * *

W artykule zostały przedstawione tylko niektóre spośród dość licznych złożonych języków programowania robotów.

Szczególną aprobatę zyskało podejście do programowania robotów zastosowane w ALU (a w efekcie — wiele naśladownictw), dlatego też język ten został opisany bardziej szczegółowo.

Dla dopełnienia obrazu należy jednak nadmienić, że stosowane są inne podejścia do języków programowania wysokiego poziomu. Przykładowo — u podstaw języka RAPT (Robot APT), opracowanego w University of Edinburgh, leży język do sterowania obrabiarek APT (Automatically Programmed Tools). Autorzy tego języka wyszli z założenia, że robot przemysłowy najczęściej będzie współpracował z gniazdami obróbczymi sterowanymi numerycznie. Dlatego też opis obiektów nie powinien się różnić od tego, jaki jest stosowany do programowania obrabiarek. Strukturę programu podzielono na trzy części. Pierwsza to opisy — definicje przedmiotów, a kolejne to operacje, jakie robot ma wykonać (np. otwieranie chwytaka lub ruch ramienia) oraz sytuacje, jakie powinny zaistnieć w trakcie montażu (np. w przypadku przykręcania kątownika do belki — kątownik musi leżeć na belce w ten sposób, aby otwory w kątowniku i w belce były współśrodkowe). Wykonanie programu polega na przeprowadzaniu operacji oraz uzyskaniu poszczególnych sytuacji.

Do złożonych języków programowania zaliczamy również te, które są ukierunkowane na wykonanie zadania. Ponieważ w większości są to powszechnie znane języki sztucznej inteligencji, nie będziemy się tu nimi zajmować, a jedynie ograniczymy się do ich wymienienia. Najpopularnejsze z nch to różne odmiany LISPU, SAIL, POP-2, AUTOPASS, MICROPLANNER oraz CONNIVER.

LITERATURA

[1] Blume C.: PASRO. BIOMATIK informationen, 1983
[2] Gini G., Gini M.: Object description with a manipulator. The Industrial Robot, Vol. 5, No. 1, 1978
[3] Gini G., Gini M.: Using a task description language for assembly. The generation of world models. Proceedings of the 8th International Symposium on Industrial Robots, Vol. 1, 1978
[4] Luh J.: An Anatomy of Industrial Robots and Their Controls. IEEE Transactions on Automatic Control, Vol. AC-28, No. 2, February 1983
[5] Mujtaba S., Goldman R.: AL User's Manual. Stanford Artificial Intelligence Laboratory, 1979
[6] Paul R.: Cartesian Coordinate Control of Robots in Joint Coordinates. 3rd Symposium on Theory and Practice of Robots and Manipulators 1978, PWN, 1980
[7] Popplestore R., Ambler A., Bellos I.: RAPT: A language for describing assemblies. The Industrial Robot, September 1978
[8] Popplestore R., Ambler A., Bellos I., Sheeler M.: Formation of body models and their use in robotics. Proceedings of the 8th ISIR, Vol. 1, 1978
[9] Rieger C., Rosenberg J., Samet H.: Artificial Intelligence Programming Languages for Computer Aided Manufacturing. IEEE Transactions on Systems, Man, and Cybernetics. Vol. SMC-9, No. 4, April 1979.

Artykuły J. L. Lauriere'a — „Knowledge representation and use" (TSI, vol. 1, no 1, 2, 1983) — stanowią przegląd najbardziej znanych systemów ekspertowych i technik ich programowania. Poniższe opracowanie przedstawia system MYCIN, który — jak dotychczas — jest jednym z najpełniej i najlepiej zrealizowanych systemów ekspertowych. Czytelnicy głębiej zainteresowani problematyką reprezentacji i automatycznego przetwarzania wiedzy oraz przeglądem istniejących systemów ekspertowych muszą, niestety, poszukać oryginalnych artykułów opublikowanych w TSI. (Red.)

MYCIN — wzorcowy system ekspertowy

We wczesnych badaniach związanych ze sztuczną inteligencją często stawiano sobie zbyt wygórowane cele. Dążono do automatycznego rozwiązywania problemów obejmujących ogromne zasoby wiedzy. W ostatnich latach znacznie rozwinęły się tzw. systemy ekspertowe (ang. expert systems), przed którymi stawiane jest skromniejsze, choć nadal bardzo ambitne zadanie: osiągnięcia poziomu wiedzy najlepszych specjalistów w dobrze określonej, lecz wąskiej dziedzinie.

Systemy ekspertowe powinny mieć oczywiście możliwość gromadzenia i poszerzania wiedzy oraz zarządzania nią. Istotne jest, iż tworzone są one d l a l u d z i. Konieczne jest zatem, aby mogły prowadzić ze specjalistami dialog w języku naturalnym, a w razie wystąpienia jakichkolwiek wątpliwości — tłumaczyć i motywować swoje rozumowanie. Ważne jest ponadto umożliwienie łatwego przekazywania systemowi wiedzy przez specjalistów: system powinien tę wiedzę analizować, porównywać ją z dotychczasową i ewentualnie rozszerzać swe umiejętności.

MYCIN-TEIRESIAS

Opiszemy teraz system MYCIN [5], który wraz z programami TEIRESIAS [2] i BAOBAB [1] tworzy typowy system ekspertowy, umożliwiający posługiwanie się wiedzą reprezentowaną w formie deklaracyjnej. TEIRESIAS służy do midyfikacji aktualnego zasobu wiedzy systemu, zaś BAOBAB odpowiedzialny jest za konwersację w języku naturalnym.

MYCIN powstał z myślą o wspomaganiu pracy lekarzy (diagnostyki i leczenia) w przypadkach zakażenia krwi. Konwersacja systemu z lekarzem toczy się w języku angielskim. Na podstawie niepełnych najczęściej informacji o stanie pacjenta oraz o wynikach testów laboratoryjnych, MYCIN stawia diagnozę i proponuje alternatywne sposoby leczenia. W każdym momencie system może wyjaśniać swe rozumowanie i odpowiadać na pytania.
Stan wiedzy systemu zmienia się dynamicznie.

Techniki użyte przy tworzeniu MYCINA zostały wykorzystane także w innych systemach ekspertowych, wspomagających:

— naprawę samochodów
— leczenie zapalenia opon mózgowych
— diagnozę chorób płucnych
— poszukiwania minerałów
— inżynierię genetyczną.

Warto zauważyć, że problem diagnozy ograniczony nawet do szczególnej, wąskiej dziedziny, jak w przypadku zakażenia krwi, jest bardzo złożony, gdyż:
— nie istnieje uniwersalny lek na wszystkie rodzaje zakażeń
— należy automatycznie ocenić na ile poważna jest infekcja
— należy stwierdzić jaki rodzaj bakterii wywołał zakażenie
(Jest to poważny problem — diagnostyka laboratoryjna wymaga 48 godzin dla podania precyzyjnej charakterystyki. Czas ten jest zbyt długi w przypadkach wymagających szybkiego podjęcia leczenia, zatem system musi działać w warunkach niepełnej informacji.)
— należy uwzględniać różnice pomiędzy pacjentami (alergie, ciśnienie krwi, wiek, inne choroby itp.).

Zobaczymy dalej, że reprezentacja wiedzy w formie deklaracyjnej wraz z odpowiednimi regułami wnioskowania [1]) pasuje znakomicie do rozwiązywania podobnych problemów co więcej — umożliwia dialog w języku naturalnym.

BUDOWA SYSTEMU

System MYCIN zbudowany jest z bazy danych reprezentujących wiedzę o zakażeniach krwi. Schemat systemu przedstawiony jest na rysunku 1.

Głównym źródłem wiedzy są reguły wnioskowania (baza danych zawiera ich ponad dwieście). Schemat każdej z reguł jest następujący:

if założenie 1
and if założenie 2
.....
and if założenie k
then wniosek (akcja, którą należy podjąć jeśli założenia są spełnione).

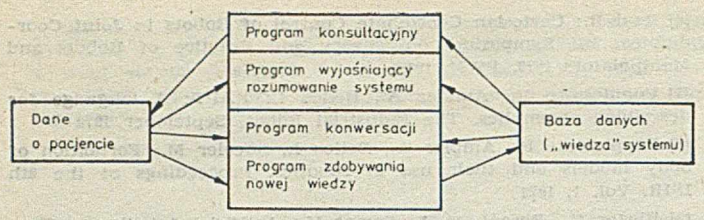
Na przykład — regułą jest:

MYCIN, Reguła 217

if 1. bakterie rozpoznano jako Bacteriodes
and if 2. podłoże hodowli bakterii było sterylne
then proponowana terapia powinna być wybrana spośród następujących leków: chloramphenicol, clindamycin, tetracycline, lindomycin, gentamycin.

---

[1]) Angielski termin „production rules" pozwoliłem sobie nieprecyzyjnie, choć — jak sądzę — adekwatnie przetłumaczyć na polskie r e g u ł y w n i o s k o w a n i a — przyp. A.S.



Rys. 1. Struktura systemu MYCIN

System MYCIN zbudowany jest za pomocą dwudziestu czterech standardowych predykatów (identyczność, zawieranie itd.), jedenastu obiektów (bakteria, hodowla, lek itd.) oraz osiemdziesięciu atrybutów identyfikujących bakterie, podłoża hodowli itp. Reguły wprowadzane są do systemu w języku naturalnym.

Z akcjami (wnioskami) związane są współczynniki wiarygodności (WW), będące liczbami rzeczywistymi z przedziału [—1, 1]. Ich wprowadzenie było niezbędne, ponieważ wiedza systemu nie ma charakteru absolutnego; jeśli np. nie ma pełnej informacji o stanie zdrowia pacjenta, to rozumowanie systemu nie może prowadzić do wniosków pewnych.

Współczynnik wiarogodności nowych faktów wywnioskowanych na podstawie aktualnego stanu wiedzy systemu wyliczany jest zgodnie z zależnością:

$$WW = RM * \min WW_i$$

gdzie **RW** jest współczynnikiem wiarogodności użytej reguły, a WW_i — współczynnikiem wiarogodności **i**-tej przesłanki tej reguły.

Jeśli podczas rozumowania dwie reguły dały te same wnioski, ze współczynnikami wiarogodności **C1** i **C2**, to przyjmuje się, że:

$$WW_{wniosku} = C1 + C2 - C1 * C2$$

Każdy fakt, dla którego wartość bezwzględna **WW** jest mniejsza od 0,2 jest eliminowany z bazy danych przez system.

Powyższa metoda wyliczania **WW** była krytykowana np. w pracach [3, 4, 6]. Tam też znaleźć można inne propozycje obliczania **WW.**

SYSTEM WNIOSKOWANIA

Reguły wnioskowania przedstawić można w formie:
$LS \rightarrow PS$

gdzie lewa strona **(LS)** opisuje szczególną sytuację, zaś prawa (PS) — akcję, która powinna zostać podjęta w tej sytuacji.

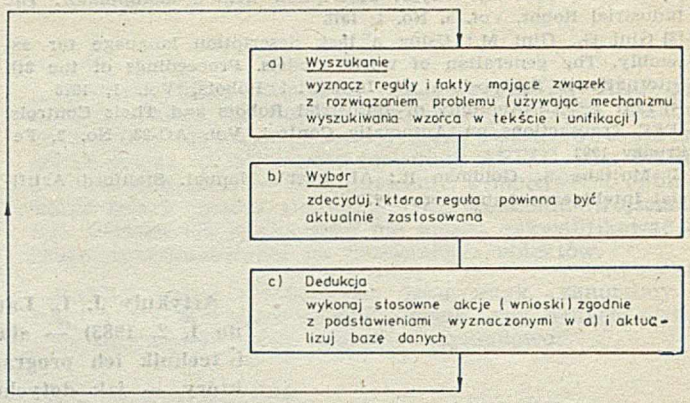
Rozważmy pewien przykład wnioskowania. Załóżmy, że wiedzę reprezentują reguły:

R1. $A \rightarrow E$
R2. $B \rightarrow D$
R3. $H \rightarrow A$
R4. $E_\Delta G \rightarrow C$
R5. $E_\Delta K \rightarrow B$
R6. $D_\Delta E_\Delta K \rightarrow C$
R7. $G_\Delta K_\Delta F \rightarrow A$

oraz że początkowa baza danych zawiera fakty **H** oraz **K.** Na podstawie reguł R1—R7 oraz faktów **H** i **K** wnioskujemy:

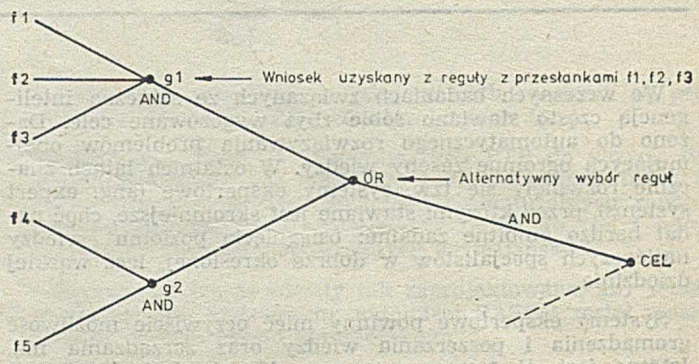
$$H \xrightarrow{R3} A,\ A \xrightarrow{R1} E,\ E_\Delta\ K \xrightarrow{R5} B,\ B \xrightarrow{R2} D,\ D_\Delta\ E_\Delta\ K \xrightarrow{R6} C$$

W skład systemu wnioskowania wchodzi:
— zbiór reguł (odpowiadający tzw. długoterminowej pamięci człowieka)
— robocza baza danych, zawierająca wszystkie fakty, które program mógł wywnioskować w danym momencie (odpowiada ona krótkoterminowej pamięci człowieka)
— interpreter odgrywający kluczową rolę w procesie dedukcji (schemat jego pracy przedstawiony jest na rysunku 2).



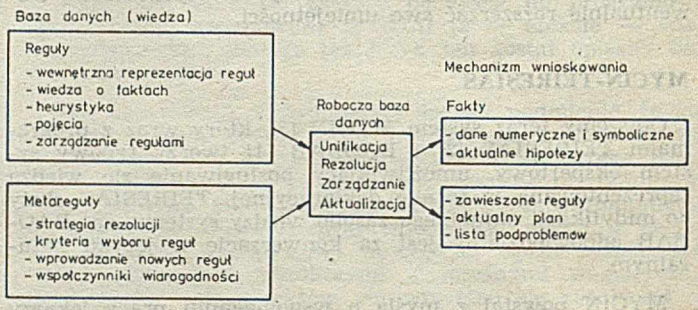
Rys. 2. Podstawowy cykl pracy interpretera

Początkowo robocza baza danych zawiera jedynie problem do rozwiązania (cel). Cel osiągany jest podobnie jak przy dowodzeniu twierdzeń. Użyta tu metoda może być przedstawiona za pomocą drzewa **AND-OR** (por. rys. 3). Węzły **OR** tego drzewa odpowiadają wyborom różnych reguł, węzły **AND** — przesłankom wybranych reguł.



Rys. 3. Drzewo wnioskowania

Ogólny schemat budowy typowego systemu wnioskowania przedstawia rysunek 4.



Rys. 4. Ogólny schemat budowy typowego systemu wnioskowania

W systemie MYCIN rozumowanie rozpoczyna się od rozpoznawania bakterii, które mogły wywołać zakażenie. Konstruowane jest drzewo AND-OR. Jeśli cel został osiągnięty, system rozważa wszystkie reguły, które doprowadziły do rozwiązania. Lewa strona reguły (węzeł **AND**) służy do wyznaczania współczynnika wiarogodności **r.** Jeśli $|r| \geqslant 0{,}2$, to formułowana jest prawa strona reguły oraz wyliczany dla niej WW. Jeśli $|r| < 0{,}2$, to reguła jest odrzucana. W przypadku, gdy system na podstawie aktualnej wiedzy o pacjencie nie jest w stanie wyciągnąć żadnego wniosku, zadaje on dodatkowe pytania.

Dokończenie na str. 21

W krajach zachodnich mikrokomputery nie są w zasadzie wykorzystywane do... wykonywania obliczeń. O karierze komputera osobistego przesądziły zastosowania w dziedzinach dalekich od wyliczania całek, różniczek itp. Procesor tekstowy czy też programy „biurowe" nie wychodzą z reguły poza podstawowe cztery działania. Dlatego też w podawanych parametrach mikrokomputera niezmiernie rzadko spotyka się dokładność numeryczną obliczeń.

Z kontaktów z czytelnikami wynika, że krajowi użytkownicy mikrokomputerów dzielą się na fanatyków rozrywki (gry komputerowe) i sfrustrowanych naukowców, którzy — zniechęceni sposobem funkcjonowania naszych ośrodków obliczeniowych — pragną poprawić sobie nieco warunki pracy, choćby kosztem spożywania suchego chleba na stypendiach zagranicznych. Dlatego też opracowanie dobrych testów numerycznych wydaje się być — z naszej polskiej perspektywy — sprawą dość istotną. Można by wprawdzie zalecić korzystanie ze sprzętu firm specjalizujących się w obsłudze naukowców (np. HEWLETT PACKARD), ale biorąc pod uwagę jego ceny — wielu jajogłowych przypłaciłoby to zagłodzeniem. Zapewne los głodujących na Zachodzie naukowców (naszych) wzruszy Czytelników, oczekujemy więc na propozycje ciekawych testów dokładności numerycznej.

Jaka dokładność numeryczna

Zamieszczone w mikroKLANIE 3 programy testujące nie pozwalały na porównanie dokłaności numerycznej komputerów. Jednak w wielu zastosowaniach właśnie dokładność numeryczna, a nie czas obliczeń, odgrywa zasadniczą rolę. Poniżej prezentujemy tekst dokładności obliczeniowej (pierwiastek kwadratowy, kwadrat liczby oraz generator liczb losowych) i jego wyniki dla popularnych komputerów osobistych. Jak poprzednio, zostawiamy wolne miejsce w tabeli na testy innych komputerów, posiadanych przez Czytelników. A może ktoś zaproponuje uniwersalny test wszystkich funkcji numerycznych dla języka BASIC?

```
10 R=0
20 S=0
30 FOR N=1 TO 100
40 A=N
50 FOR I=1 TO 10
60 A=SQR (A)
70 R=R+RND (1)
80 NEXT I
90 FOR I=1 TO 10
100 A=A**2
110 R=R+RND (1)
120 NEXT I
130 S=S+A
140 NEXT N
150 PRINT ABS (1010—S/5)
160 PRINT ABS (1000-R)
170 END
```

Porównanie parametrów dokładności numerycznej najbardziej popularnych modeli komputerów osobistych

Komputer	Czas[1])	Dokładność	Losowość
BBC Acorn	0 : 21	0,0000128746033	5,2
IBM PC	0 : 24	0,01159668	6,3
TRS 80/4	0 : 53	0,0670776	6,5
HP 85A	1 : 20	0,00000002	5,8
HP 75C	1 : 38	0,00000002	5,8
VIC 20	1 : 49	0,0010414235	23,7
CBM 64	1 : 53	0,0010414235	8,9
Apple IIe	1 : 53	0,0010414235	12,0
TI 99/4A	3 : 46	0,00000011	2,6
ZX81	4 : 23	0,0006685257	6,3
SPEC-TRUM	4 ; 39	0,0006685257	3,5
HP 41C	31 ; 48	0,000034000	7,8[2])
,,,,,,,,,,,,,	,,,,,,,,,	,,,,,,,,,,,,,,,,,,,,	,,,,,,,,,

1) minuty : sekundy
2) generator liczb losowych z PPC ROM, funkcja RN

Wyniki testów: przypisujemy wartości 0,0000001 dokładność bardzo dobrą, wartościom powyżej 0,01 — złą; podobnie losowość poniżej 10 jest dobra, powyżej 10 — zła.

Opracował
JAKUB TATARKIEWICZ
na podstawie „Creative Computing", styczeń 1984

Strach przed ?

Amerykańskie pismo POPULAR COMPUTING, zajmujące się mikroinformatyką, opublikowało niedawno (nr 6/84) artykuł Jerry Pournelle'a o perspektywach tej dziedziny w różnych krajach. Autor twierdzi kategorycznie, że w państwach RWPG, a w tym także w PRL, mikroinformatyka nie będzie rozwijana. Wprost przeciwnie — będzie świadomie hamowana, jako zagrażająca kontroli przepływu informacji. Podobny pogląd wyraził — w ostrej propagandowo audycji — GŁOS AMERYKI.

To groźne. Bo choć nie widać poważnych argumentów na uzasadnienie takiej tezy — do powstania w Polsce niezależnej sieci informacyjnej wykorzystującej mikrokomputery jest naprawdę bardzo daleko — to jednak znajdują się one post factum. Urzędnik przestaje pytać, do czego służy komputer, a zaczyna kojarzyć nożyce ze stołem.

Do Przemysława Filipowicza z Lubartowa nadeszła w kwietniu 1984 br. przesyłka od brata z Toronto (Kanada), zawierająca mikrokomputer COMMODORE VIC-20, zasilacz, magnetofon kasetowy, komplet kaset do gier telewizyjnych i nauki języka angielskiego oraz przewody połączeniowe. Urząd Celny w Gdyni odmówił wydania pozwolenia na przywóz, stwierdzając, że:

„Urząd Celny mógłby wydać pozwolenie na przywóz przedmiotowego komputera tylko w przypadku, gdyby komputer ten był niezbędny do pracy zawodowej względnie naukowej Obywatela".

Udowodnij, żeś nie nosorożec! Obywatel wysłał zaświadczenie o wykonywanych przez siebie usługach w dziedzinie elektromechaniki samochodowej i o żonie — ekonomistce, opisał niedouczenie politechniczne dzieci (11 i 17 lat). I nic... Nosorożec! Gdynia komputera nie przepuściła.

Skończyło się jednak pozytywnie. W sierpniu (!) Główny Urząd Ceł w Warszawie uwzględnił odwołanie. Okazało się przy tym, że wprawdzie nie ma żadnych formalnych ograniczeń w przywozie komputerów (poza koniecznością uzyskania pozwolenia...), ale niedawno (ponoć w czerwcu) trochę się „zaostrzyło". Przygotowywana jest ponadto — jak powiedziano w GUC — normalizacja prawna tego rodzaju przywozów.

Niby chcemy gonić świat, nie pozostawać w zaścianku... Ale jakże tu nie dostrzegać tak rozgłaszanych zagrożeń. Więc na wszelki wypadek wzmożemy czujność. Bo — jak stwierdził kompetentny urzędnik GUC — decyduje przecież dobro państwa.

ZBIGNIEW GLUZA

- Na rynku brytyjskim pojawiły się pierwsze komputery osobiste w standardzie MSX (zaproponowanym przez MICROSOFT). Ceny w granicach 250—300 funtów. Jest to ponad dwukrotnie więcej niż w Japonii, kraju producentów! Szkoda, rozwiązania techniczne w standardzie są bardzo ciekawe — ale ta cena...

- Witold Majewski (Wrocław tel. d, 55-48-96; służb. 67-90-71 w. 203) jest posiadaczem ZX SPECTRUM i bogatego oprogramowania (m.in. ASSEMBLER, FORTH, VU- FILE, High Resolution), które chętnie udostępni innym w drodze wymiany. Poszukuje również papieru termicznego (szer. 10 cm) do drukarki.

Opisywaliśmy już sposób zrealizowania współpracy z magnetofonem kasetowym. Obecnie zamierzamy przedstawić rozwiązanie umożliwiające przyłączenie do mikrokomputera napędu dysków elastycznych. Pamięć masowa tego typu jest niewątpliwie wielokrotnie droższa, umożliwia jednak zainstalowanie jednego z popularnych systemów operacyjnych (np. CP/M) i wykorzystanie stworzonego oprogramowania do nieco poważniejszych zastosowań niż gry i zabawy.

Zanim opiszemy konkretną realizację sprzęgu dla mniej otrzaskanych Czytelników przygotowaliśmy tekst wprowadzający w zagadnienia związane z dyskami elastycznymi.

Elastyczne dyskietki

Nie istnieje komputer bez pamięci. W erze komputerów-mamutów towarzyszyły im równie wielkie (rozmiarami) i równie powolne pamięci. Nie będziemy w tym miejscu robić przeglądu historycznego — dość powiedzieć, że w ramach poszukiwań rozwiązania, które pozwoliłoby na tanie przechowywanie dużych ilości łatwo dostępnej informacji, wymyślono elastyczne dyskietki. Od momentu wprowadzenia tej techniki upłynęło już trochę czasu i komputery zdążyły na tyle stanieć, że to co kiedyś wydawało się tanie, dzisiaj wymaga często większych nakładów finansowych niż sam komputer. Mimo dalej prowadzonych na szeroką skalę poszukiwań (EAROM-y, pamięci domenowe) na razie nie ma rozwiązania, które za niższą cenę oferowałoby równie „elastyczny" sposób przechowywania informacji.

Dyskietka to krążek sztucznego tworzywa (mylar), pokrytego tlenkiem metalu (nośnik magnetyczny) i zapakowanego w kartonową kopertę chroniącą przed pyłem i uszkodzeniami.

Opracowano wiele standardów dotyczących sposobu zapisywania informacji. Dla dyskietek o średnicy 8" wykorzystuje się zazwyczaj 77 ścieżek, dzielonych na sektory. Istnieją dwie metody wydzielania sektorów na ścieżce. Jedna z nich — mechaniczna (ang. hard-sectors) — polega na wyznaczaniu sektorów przez specjalnie wykonane w dyskietce otwory. W drugiej — na ścieżce zapisuje się specjalną informację wydzielającą poszczególne sektory (ang. soft-sectors). Metoda mechaniczna pozwala na zapis większej ilości informacji, narzuca jednak określoną długość sektora. Druga metoda, choć bardziej rozrzutna, zdobyła większą popularność, gdyż wykorzystywana jest przez potentata informatycznego — IBM.

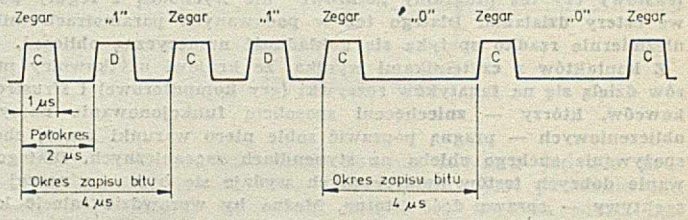
Najczęściej wykorzystywane są dwa standardowe formaty zapisywania danych:
— IBM 3740, tzw. pojedyncza gęstość (SD, ang. Single Density)
— IBM 34, tzw. podwójna gęstość (DD, ang. Double Density).

W zależności od konstrukcji używanego napędu dyskietek, informacja może być zapisywana tylko z jednej strony dyskietki (SS, ang. Single Side) lub z obydwu (DS, ang. Double Side) — w tym drugim przypadku konieczne są dwie głowice zapisująco-odczytujące.

Kodowaniej dla pojedynczej gęstości

Format IBM 3740 wykorzystuje technikę modulacji częstotliwości (FM, ang. Frequency Modulation) dla zakodowania strumienia danych zapisywanego na ścieżce. W rozwiązaniu tym bity danych rozdzielane są impulsami zegarowymi, co zapewnia utrzymanie stałej synchronizacji w trakcie odczytu. Rzut oka na rysunek 1 wystarczy do zrozumienia, dlaczego ten sposób kodowania nazywany jest modulacją częstotliwości.

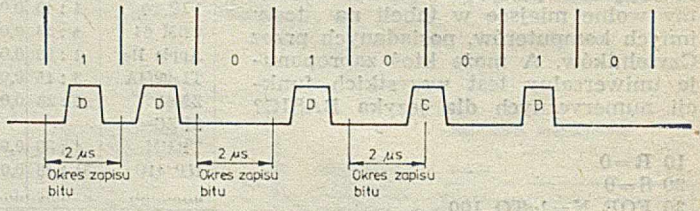
Przy odczycie informacji należy nie tylko przekształcić postać danych z szeregowej na równoległą, ale także odseparować zegar. Istnieją dwie metody separacji. Pierwsza polega na zastosowaniu generatora monostabilnego. Mimo niewielkich kosztów i prostej realizacji, nie zaleca się jej stosowania ze względu na małą odporność na zakłócenia wynikające z nierównomiernej pracy zespołów mechanicznych. Drugi sposób wymaga zastosowania układu generatora z zamkniętą pętlą fazową (PLO, ang. Phase Locked Oscilator). W przypadku kodowania z podwójną gęstością konieczne jest zastosowanie układu PLO.



Rys. 1. Sposób kodowania danych dla zapisu o pojedynczej gęstości — modulacja częstotliwości (FM)

Kodowanie dla podwójnej gęstości

Zwiększenie ilości informacji przechowywanej na dyskietce przez zagęszczenie zapisu nie jest praktykowane, gdyż stosowana gęstość bliska jest fizycznemu ograniczeniu dla stosowanego nośnika magnetycznego. Można jednak zastosować „gęstszą" metodę kodowania. Standard IBM 34 jest zmodyfikowaną metodą modulacji częstotliwości (MFM. ang. Modified Frequency Modulation). Ograniczono w nim występowanie impulsów zegarowych do sytuacji, gdy zachodzi konieczność rozdzielenia bitów zerowych. O ile bitowi „1" odpowiada zapisanie impulsu, o tyle dla bitu „0" impuls nie jest zapisywany. Dla rozgraniczenia ciągu bitów zerowych trzeba więc zapisać impulsy zegarowe. Sposób kodowania MFM został zobrazowany na rysunku 2. Ponieważ dla zmodyfikowanej modulacji częstotliwości liczba impulsów zmniejsza się o połowę, okres trwania komórki informacji można zredukować z 4 µs do 2 µs. Minimalny odstęp między impulsami w obydwu standardach wynosi 2 µs.



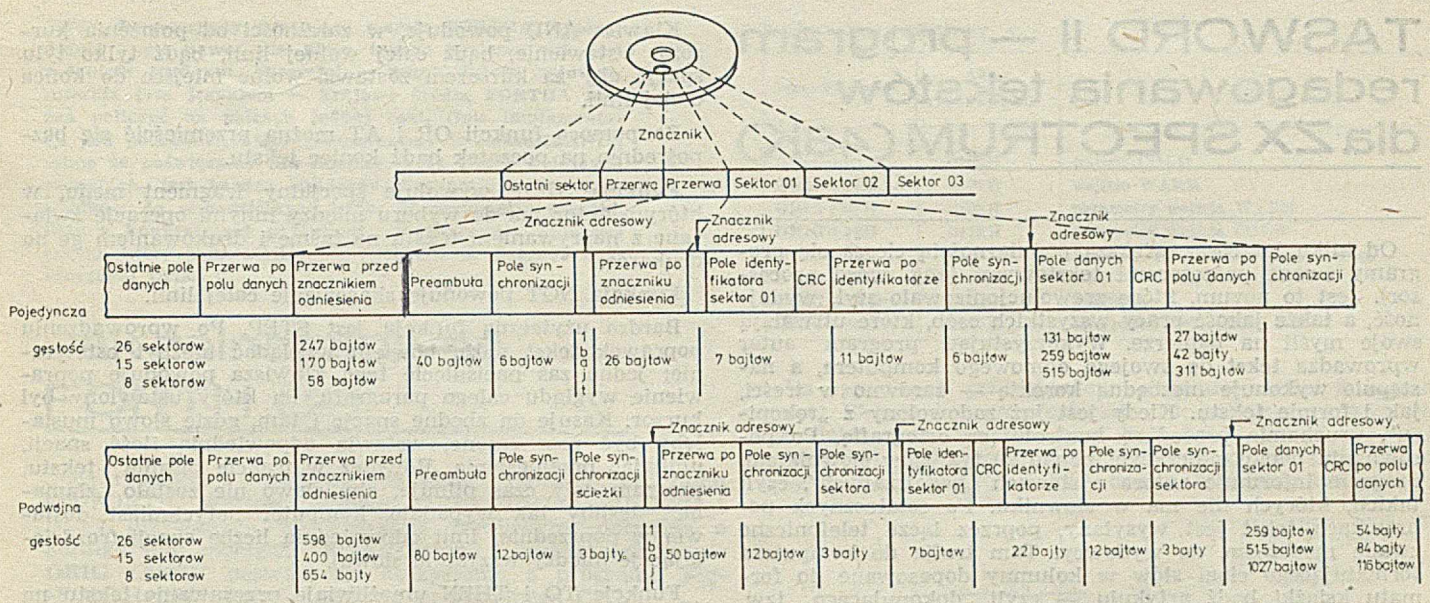
Rys. 2. Sposób kodowania danych dla zapisu o podwójnej gęstości — zmodyfikowana modulacja częstotliwości (MFM)

Organizacja zapisu informacji na ścieżce

Każda ścieżka może zostać podzielona na 26, 15 lub 8 sektorów. Podział jest dokonywany na etapie inicjowania dysku, czyli tzw. formatowania. Dyskietki są zazwyczaj sformatowane przez producenta, jednak większość systemów wspomagania projektanta oprogramowania również może realizować to zadanie. W trakcie formatowania informacja zapisana wcześniej na dyskietce zostaje skasowana.

W zależności od zastosowanego podziału ścieżki, sektor może zawierać 128, 256 lub 512 bajtów dla SD i 256, 512 lub 1024 dla DD.

Na rysunku 3 przedstawiono organizację zapisu na ścieżce. Punktem odniesienia jest fizyczny znacznik początku (otwór). Pierwszy sektor występujący za znacznikiem ma zawsze przypisany numer 1. Następne sektory mogą być ponumerowane w dowolnej kolejności. Numer sektora zapisywany jest w polu identyfikującym (ang. ID field) w trakcie formatowania dyskietki. Pola identyfikacji i pola danych są rozgraniczane przerwami (ang. gap).



Rys. 3. Struktura zapisu na ścieżce dyskietki dla zapisu o pojedynczej i podwójnej gęstości

Każde pole rozpoczyna się od sześciu bajtów składających się z samych zer (ang. sync field), po których następuje 1-bajtowy znacznik adresu (ang. address mark). Stosowane są cztery typy znaczników adresu (o różnych kodach) dla oznaczenia:

— fizycznego początku ścieżki
— początku pola identyfikującego
— początku pola danych
— początku pola skasowanych danych.

Pole identyfikujące składa się z siedmiu bajtów, w których kolejno zapisane są: znacznik adresu, numer ścieżki, numer głowicy, numer sektora, rozmiar sektora oraz dwa bajty CRC (ang. Cyclic Redundancy Check).

Pole danych składa się ze znacznika adresu, odpowiedniej liczby bajtów danych i dwóch bajtów CRC.

Sprzęg z napędem dyskietek elastycznych

Z chwilą włożenia dyskietki do kieszeni napędu i zamknięcia osłonki kończy się obsługa manualna. Dalsza współpraca z napędem realizowana jest poprzez sterownik sprzęgu.

Zasadniczym zadaniem jest ustawienie głowicy zapisująco-odczytującej nad żądaną ścieżką. Jest to realizowane przez silnik krokowy i dlatego logika sprzęgu dostarcza sygnał w rodzaju: krok w przód, krok w tył. Jeżeli rozpoczynamy współpracę z dyskietką, stosowany jest specjalny mechanizm (dioda i fotokomórka), pozwalający odnaleźć punkt odniesienia, czyli — ścieżkę zerową. Głowica ustawia się automatycznie na ścieżkę zerową — w wyniku zerowania lub wykrycia błędu pozycjonowania.

Do odnalezienia konkretnej ścieżki konieczna jest znajomość aktualnej pozycji głowicy. W tym celu sterownik napędu zazwyczaj zawiera specjalny rejestr, w którym przechowywany jest numer ścieżki, nad którą znajduje się głowica. Dzięki temu możliwe jest wygenerowanie odpowiedniej liczby kroków w przód lub do tyłu. Prawidłowość ustawienia głowicy weryfikowana jest przez porównanie z informacją zapisaną w polu identyfikującym.

Dla dokonania operacji zapisu lub odczytu niezbędne jest opuszczenie głowicy na powierzchnię dysku. Jest to tzw. ładowanie głowicy. Przed przesunięciem na inną ścieżkę głowica nie musi być podnoszona.

W sprzęgu z napędem dyskietek stosowane są również sygnały nie związane bezpośrednio z obsługą funkcji mechanicznych. Linia SELECT wykorzystywana jest do wybrania napędu lub głowicy, za pomocą której realizowana będzie operacja zapisu lub odczytu. Z kolei linia READY (sterowana przez układy elektroniczne napędu) sygnalizuje możliwość współpracy (włączone zasilanie, włożona dyskietka, zamknięta osłona itp.).

Niektóre napędy mają dodatkowo możliwość sygnalizowania, że dyskietka jest zabezpieczona przed zapisem (ang. Write Protect) oraz sygnalizację uszkodzeń układu zapisu (ang. Write Fault).

W miarę przesuwania się w kierunku osi dysku długość ścieżek maleje, co powduje większą gęstość zapisu.

Dla skompensowania tego zjawiska, od połowy dyskietki stosuje się mniejszy prąd zapisu. Linia sterująca tą funkcją nazywana bywa „LOW CURRENT" lub „TG43" (ang. track greater than 43).

W następnym numerze mikroKLANU opiszemy przykład konkretnej realizacji sprzęgu.

AJP

- Od 1 października w klubie Wydawnictwa SIGMA (Warszawa, ul. Mazowiecka 12) obok zwyczajowego spotkania hobbystów-mikroinformatyków prowadzona będzie giełda sprzętu i oprogramowania. Ponieważ na „Perskim" zabroniono tego typu działalności, organizatorzy pragną w ten sposób podtrzymać tak cenne dla hobbystów forum wymiany. Planowane jest organizowanie giełd co miesiąc. Zapraszamy wszystkich zainteresowanych, a szczególnie tych, którzy mają coś do zaoferowania innym.

- Użytkownicy mikrokomputerów MERITUM I oraz ZX-81 poszukują kontaktów z innymi posiadaczami tego sprzętu lub TRS 80/II. Szczególnie interesuje ich oprogramowanie — tak w BASICU, jak i w języku maszynowym Z80. Na razie korzystają z czasopisma BYTE; będą zobowiązani za udostępnienie innych tytułów z gotowymi programami. Kontakt: Politechnika Łódzka, centr. tel. 36-55-22; Zbigniew Chrześcijański — tel. w. 673, Jacek Żebrowski — tel. w. 674; ew. wiadomość na tel. bezp. w Łodzi — 36-76-44.

TASWORD II — program redagowania tekstów dla ZX SPECTRUM (48K)

Od kilku lat coraz większą popularnością cieszą się programy zwane procesorami tekstowymi (ang. word processor). Jest to novum, które zrewolucjonizowało styl, wydajność, a także jakość pracy wszystkich osób, które utrwalają swoje myśli na papierze. Wykorzystując program autor wprowadza tekst do swojego domowego komputera, a następnie wykonuje niezbędną korektę — zarówno w treści, jak i formie tekstu. Kiedy jest już zadowolony z „rękopisu", uruchamia procedurę kontrolującą ortografię. Po porównaniu tekstu z liczącym ponad 20 tys. słów słownikiem, program informuje autora o słowach „podejrzanych", czyli takich, których nie ma w słowniku. Po ostatecznych poprawkach tekst jest wysyłany, poprzez łącze telefoniczne zwane modemem, do wydawcy. Tam trafia do komputera formatującego ciągi słów w kolumny dopasowane do formatu książki bądź artykułu — czyli dokonującego tzw. składu. Po czym, zależnie już tylko od wydajności drukarni, można rozpocząć kolportaż wydrukowanego tekstu, ukończonego przez autora kilka dni wcześniej.

Najciekawsza cecha programu TASWORD II dla ZX SPECTRUM jest to , ze na jednej linii potrafi on zmiescic 64 znaki. Kazdy kto pracowal z ZX SPECTRUM na pewno narzekal na mala (32) liczbe znakow w linii. Omawiany program, wykorzystujac specjalne procedury napisane w jezyku maszynowym, drukuje znaki o szerokosci czterech punktow. Wbrew pozorom sa one calkiem czytelne. 1]

Gdyby jednak ktoś upierał się, że litery są za mało czytelne, zawsze może „włączyć" opcję **c**, która pokazuje na ekranie aktualnie poprawiany bądź wpisywany wycinek tekstu, przy wykorzystaniu standardowej wielkości liter.

Aby autorowi opłacało się wykorzystywać omawiany program, musi on w istotny sposób ułatwiać prace. Służy temu cała gama specjalnych funkcji.

Jedną z najbardziej użytecznych jest EDIT. Po wprowadzeniu jej na ekranie pojawia się strona pomocnicza, na której w wielkim skrócie opisane są wszystkie możliwe polecenia (tzw. menu). Po naciśnięciu obydwu klawiszy SHIFT, wyświetlane są te polecenia, których zastosowanie wymaga równoczesnego naciśnięcia ich obu.

Klawisze 3 i 4, czyli TRUE VIDEO i INV. VIDEO, służą do szybkiego przenoszenia kursora przez tekst. Szybkość jest uzyskiwana dzięki temu, że kursor zatrzymuje się tylko na początku każdego słowa.

Klawisze 5, 6, 7 i 8 służą do przesuwania kursora w dowolne miejsce ekranu.

Klawiszem 9 uruchamia się opcje wstawiania w tekst specjalnych znaków sterujących przeznaczonych dla drukarki. Znaczenie tych znaków może zaprogramować sam użytkownik.

Klawisz <> powoduje równomierne rozrzucanie tekstu w linii, w której znajduje się kursor.

Klawisz AND powoduje, w zależności od położenia kursora, wstawienie, bądź całej wolnej linii, bądź tylko tylu spacji, aby za kursorem zostawić wolne miejsce do końca danej linii.

Za pomocą funkcji OR i AT można przemieścić się bezpośrednio na początek bądź koniec tekstu.

Funkcja STOP wywołuje specjalny fragment menu, w którym autor ma do wyboru między innymi operacje związane z nagrywaniem tekstu na taśmę i drukowaniem go na drukarce.

Funkcja NOT powoduje skasowanie całej linii.

Bardzo użyteczną funkcją jest STEP. Po wprowadzeniu poprawek tekst może bowiem wyglądać niezbyt estetycznie; jedno zaś naciśnięcie tego klawisza powoduje poprawienie wyglądu całego paragrafu, na który ustawiony był kursor. Kasuje on zbędne spacje i tam, gdzie słowo musiałoby być przenoszone, wstawia odpowiednią ilość spacji, unikając przeniesienia. Również w trakcie pisania tekstu, program cały czas pilnuje, aby słowo nie zostało „złamane". Jeśliby tak wypadało, komputer natychmiast dostawia w poprzedniej linii odpowiednią liczbę spacji (rozdzielając je między wszystkie słowa).

Funkcje TO i THEN umożliwiają przesuwanie tekstu na ekranie w górę lub w dół.

Klawisz ENTER powoduje, oczywiście, powrót karetki i przejście do następnej linii.

Druga grupa funkcji to te rzadziej potrzebne, dostępne poprzez „podwójny" SHIFT.

F i **G** powodują szybkie (22 linie na jedno naciśnięcie) przesuwanie tekstu na ekranie.

P, L i **K** związane są z obsługą drukarki ZX-PRINTER. Służą do wstawiania do tekstu znaczników dużego druku i uruchamiania procesu wydruku.

Bardzo użyteczną funkcją jest R. Umożliwia wyszukanie bądź wyszukanie i zamianę dowolnego tekstu. Niestety, nie działają one zbyt szybko i w przypadku podstawiania trzeba czekać czasem nawet ponad minutę. Ale komputer nie pozwala użytkownikowi na nudę. Cały czas na ekranie widać jak dostawia wybrany przez nas ciąg znaków, porządkując tekst po każdym dostawieniu — aby utrzymać marginesy.

Klawisz **X** służy do kasowania całego pliku. Jest on zabezpieczony przed przypadkowym użyciem.

E włącza i wyłącza rozciąganie tekstu do prawego marginesu w czasie pisania.

J powoduje rozciągnięcie bieżącej linii na całą szerokość ekranu.

H powoduje usunięcie skutków działania rozkazu **J**.

Klawisze **A, S** i **D** służą do ustalania lewego i prawego tabulatora tekstu, ewentualnie kasowania tabulatorów w ogóle.

B, V, N i **M** służą do operowania całymi blokami tekstu.

Autor może zaznaczyć początek i koniec pewnego bloku, po czym skopiować bądź przenieść ten blok w inne miejsce w tekście. Komputer, oczywiście pilnuje estetyki tekstu.

Program pozostawia w pamięci miejsce na tekst o maksymalnej długości 320 linii po 64 znaki. Odpowiada to mniej więcej jedenastu stronom maszynopisu.

Dużą zaletą programu jest fakt, iż współpracuje on z większością popularnych przystawek, takich jak KEMPSTON lub LPRINT III. Gorąco polecam ten program ludziom, którzy gdy pomyślą o przepisywaniu na maszynie, wolą w ogóle nie pisać. Jestem jednym z nich i gdyby nie program TASWORD II, to artykuł ten nigdy by nie powstał.

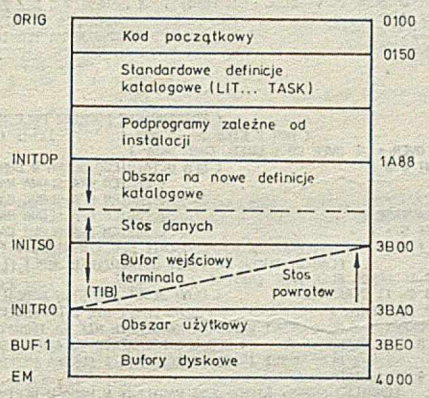
PIOTR PARLEWICZ
Warszawa

[1] Artykuł został nadesłany w takiej postaci, jak przedstawiliśmy w niniejszym akapicie (efekt działania programu TASWORD II); Nie przedstawiamy całego artykułu w tej formie, gdyż brak liter specyficznych dla polskiego alfabetu utrudniłby lekturę — przyp. red.

Gdy dysponujemy pracującą wersją FORTHA, sporządzenie kopii nawet dla nieco innego systemu nie jest zadaniem trudnym. Niemniej — mimo dużego zainteresowania tym językiem — krajowe źródła FORTHA można policzyć na palcach jednej ręki. Opis implementacji „od początku", który zamieszczamy w mikroKLANIE, mimo że poświęcamy mu miejsce w aż trzech kolejnych numerach, jest bardzo lapidarny. Zachęcamy więc Czytelników do samodzielnych prób. W przypadku porażki, listy z konkretnymi pytaniami przekażemy autorowi tekstu, który zainstalował Fig-FORTHA w systemie operacyjnym CP/M.

Jak zainstalować FORTH (2)

Na rysunku 1 przedstawiono mapę pamięci programu Fig-FORTH. Wartości symboli ORIG, INITSO, INITRO i BUF1 (używanych w programie źródłowym) decydują o konfiguracji pamięci. Przedstawione przyporządkowanie **ORIG = 0100H** pozwala na korzystanie z programu Fig-FORTH pod kontrolą systemu operacyjnego CP/M.



Rys. 1. Mapa pamięci programu fig-FORTH

Kod początkowy (tab. 1), oprócz interpretera wewnętrznego NEXT, zawiera program DEBUGGER (praca krokowa) oraz parametry i wskaźniki używane w programie. Poprzez zmianę ich wartości można modyfikować konfigurację systemu w trakcie inicjacji. Może to nastąpić bezpośrednio po włączeniu zasilania (wejście COLD) lub w środku sesji (wejście WARM).

Wykonując słowo VLIST, można uzyskać listę nazw wszystkich standardowych definicji katalogowych (wydruk 1).

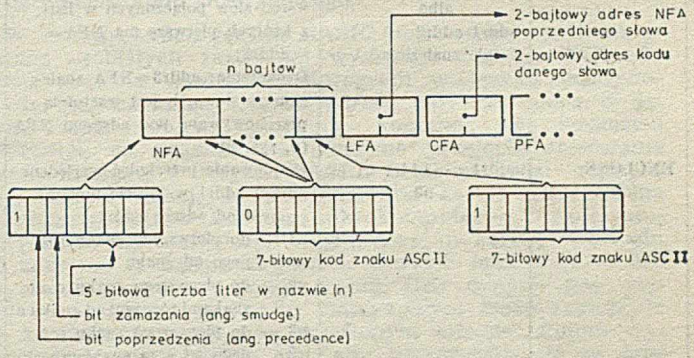
```
VLIST
TASK  .CPU  TRIAD  INDEX  LIST  BYE  VLIST  U.  ?  .  D.
.R  D.R  $S  $  SIGN  #>  <#  SPACES  WHILE  ELSE  IF
REPEAT  AGAIN  END  UNTIL  +LOOP  LOOP  DO  THEN  ENDIF
BEGIN  BACK  FORGET  '  -->  LOAD  FLUSH  R/W  SEC-WRITE
SEC-READ  TASCALC  SET-DRIVE  SET-ID  BLOCK  BUFFER  DR1
DR0  EMPTY-BUFFERS  UPDATE  +BUF  DISK-ERROR  DENSITY  #BUFF
SEC/BLK  PREV  USE  TRACK  SEC  DRIVE  P!  PG  MESSAGE
.LINE  (LINE)  M/MOD  */  */MOD  MOD  /  /MOD  M  R/  R#
MAX  MIN  DABS  ABS  D+-  +-  S->D  COLD  WARM  ABORT
QUIT  (  DEFINITIONS  FORTH  VOCABULARY  IMMEDIATE  INTERPRET
?STACK  DLITERAL  LITERAL  [COMPILE]  CREATE  ID.  ERROR
(ABORT)  -FIND  NUMBER  (NUMBER)  WORD  PAD  HOLD  BLANKS
ERASE  FILL  QUERY  EXPECT  ."  (.")  -TRAILING  TYPE
COUNT  DOES>  <BUILDS  ;CODE  (;CODE)  DECIMAL  HEX  SMUDGE
]  [  COMPILE  ?LOADING  ?CSP  ?PAIRS  ?EXEC  ?COMP  ?ERROR
!CSP  PFA  NFA  CFA  LFA  LATEST  TRAVERSE  -DUP  SPACE
ROT  >  U<  <  =  -  C,  ,  ALLOT  HERE  2+  1+  HLD
R#  CSP  FLD  DPL  BASE  STATE  CURRENT  CONTEXT  OFFSET
SCR  OUT  IN  BLK  VOC-LINK  DP  FENCE  WARNING  WIDTH
TIB  R0  S0  +ORIGIN  B/SCR  B/BUF  LIMIT  FIRST  C/L  BL
3  2  1  0  USER  VARIABLE  CONSTANT  NOOP  -1  1  2!
C!  !  2@  C@  @  TOGGLE  +!  2DUP  DUP  SWAP  DROP  OVER
MINUS  MINUS  D+  +  0<  0=  R  R>  >R  LEAVE  #S  R#!
RP@  SP!  SP@  XOR  OR  AND  U/  U*  CMOVE  CR  ?TERMINAL
KEY  EMIT  ENCLOSE  (FIND)  DIGIT  I  (DO)  (+LOOP)  (LOOP)
0BRANCH  BRANCH  EXECUTE  LIT  0K
```

Wydruk 1

Tabela 1. Kod początkowy

Adres względny	Adres bezwzględny	Opis
ORIG+00H	0100H	wejście COLD
ORIG+04H	0104H	wejście WARM
ORIG+08H	0108H	parametry wejścia WARM
ORIG+12H	0112H	parametry wejścia COLD
ORIG+26H	0126H	adres początku obszaru użytkowego
ORIG+28H	0128H	wskaźnik stosu powrotów
ORIG+2AH	012AH	debugger
ORIG+45H	0145H	interpreter wewnętrzny NEXT

Na rysunku 2 przedstawiono strukturę definicji katalogowej. Składa się ona z czterech pól: pola nazwy (ang. name field), pola łącznika (ang. link field), pola kodu (ang. code field) i pola parametrów (ang. parameter field). Symbole NFA, LFA, CFA, i PFA oznaczają adresy tych pól.



Rys. 2. Struktura definicji katalogowej

Tabela 2 przedstawia podział definicji katalogowych na grupy. Później definiowane słowa składają się na ogół z sekwencji wywołań słów zdefiniowanych wcześniej. Dlatego też procedury napisane w kodzie maszynowym występują na samym początku. Tabela 3 zawiera opis pomocniczych słów w kodzie maszynowym, nie omówionych dotąd w INFORMATYCE. Oprócz odwołań „w tył", zdarzają się także odwołania „w przód" do słów występujących później (ang. forward definitions). W dalszej kolejności występują słowa-kompilatory. Za ich pomocą można definiować nowe słowa (m.in. stałe, zmienne użytkowe i procedury w kodzie nizanym).

Tabela 2. Standardowe definicje katalogowe

Nazwy słów	Opis
LIT ,,,2! : ; CONSTANT VARIABLE USER	procedury w kodzie maszynowym kompilatory
0 ... B/SCR	stałe
+ORIGIN	specjalna procedura
SO ... HLD	zmienne użytkowe
1+ ... P!	procedury w kodzie nizanym
DRIVE ... -->	stałe, zmienne i procedury do współpracy z dyskami
, FORGET BACK BEGIN ... WHILE SPACES ... TASK	specjalne procedury instrukcje strukturalne

Stałe programu (wydruk 2), zdefiniowane przy użyciu kompilatora CONSTANT, umożliwiają odwołania przez nazwę do najczęściej używanych liczb (zamiast pośredni-

ctwa słowa LIT), skracając program w kodzie nizanym. Definiują też pewne wartości zależne od implementacji, dając możliwość łatwej ich zmiany.

Tabela 3. Niektóre procedury pomocnicze w kodzie maszynowym

Nazwa słowa	Stos	Opis
EXECUTE	addr→	wykonanie słowa, w którym CFA = addr
DIGIT	c n1→n2 1 (konwersja możliwa) albo c n1→ 0 (konwersja niemożliwa)	zamiana znaku ASCII c zgodnie z podstawą n1 na liczbę n2
(FIND)	addr1 addr2→ addr3 b 1 (znaleziono) albo addr1 addr2→0 (nie znaleziono)	poszukiwanie słowa identycznego z tekstem umieszczonym od adresu addr1 (zakończonego spacją) wśród słów połączonych w listę, z których pierwsze ma NFA= =addr2; Oznaczenia; addr3=NFA znalezionego słowa, b jest wartością przechowywaną pod adresem NFA (p. rys. 2)
ENCLOSE	addr1 c→addr1 n1 n2 n3	znajdowanie przesunięć względem adresu addr1 w tekście umieszczonym od adresu addr1: n1 — do pierwszego znaku różnego od znaku c n2 — do pierwszego znaku c za znakami różnymi od znaku c n3 — do pierwszego znaku różnego od c za znakiem o przesunięciu n2
CMOVE	addr1 addr2 n→	przepisanie n bajtów z obszaru (addr1; addr1+n) w obszar (addr2; addr2+n)
TOGGLE	addr b→	zmiana wartości tych bitów w bajcie umieszczonym pod adresem addr, którym w masce b odpowiada „1"

```
HEX
  00 CONSTANT 0
  01 CONSTANT 1
  02 CONSTANT 2
  03 CONSTANT 3
  20 CONSTANT BL    ( SPACJA)
  50 CONSTANT C/L   ( ILOSC ZNAKOW W LINII = 80D)
3BE0 CONSTANT FIRST ( ADRES PIERWSZEGO BUFORA DYSKOWEGO)
4000 CONSTANT LIMIT ( ADRES OSTATNIEGO BAJTU PAMIECI+1)
  80 CONSTANT B/BUF ( ILOSC BAJTOW DANYCH W BUFORZE DYSKOWYM-128D)
   8 CONSTANT B/SCR ( ILOSC BUFOROW W KADRZE = 8D)
```

Wydruk 2

Zmienne programu (wydruk 3) pamiętane są w obszarze użytkowym (rys. 1.). Podczas definiowania, z wykorzystaniem słowa USER, podaje się adres względny w ramach tego obszaru. Przy inicjacji przez wejście COLD niektórym z tych zmiennych nadawane są wartości początkowe.

Do definiowania słów w kodzie nizanym służy para kompilatorów: „:" i „;". W kodzie nizanym zapisana jest większość słów FORTHA, w tym także interpreter zewnętrzny. Omówienie implementacji ciekawszych słów FORTHA odłożymy do trzeciej części artykułu, poprzestając teraz na przytoczeniu tekstu źródłowego z minimalnymi komentarzami (wydruk 4).

Tym, którzy mają dostęp do instalacji fig-FORTHA proponuję sprawdzenie przedstawionych tu definicji — poprzez wprowadzenie i wykonanie (z wyjątkiem **<BUILDS** i **DOES>**) programów z wydruków 2, 3 i 4. Oczywiście, zainstalowana wersja FORTHA zawiera już te słowa. Wprowadzając je po raz drugi, uzyskujemy coś w rodzaju niepełnej kopii systemu. Wejście do nowej kopii następuje przez wykonanie słowa QUIT (zgłoszenie następuje wtedy komunikatem „KO" zamiast „OK"). Powrót do „starej" wersji następuje po wystąpieniu jakiegokolwiek błędu. Nowa kopia jest niepełna, m. in. dlatego, że nie ma nowych procedur w kodzie maszynowym. Można by — korzystając z asemblera fig-FORTHA — zdefiniować powtórnie także i te słowa. W ten sposób możliwe jest generowanie nowych wersji FORTHA działających przy innej konfiguracji sprzętowej, a nawet na innym procesorze (przy użyciu odpowiedniego programu asemblera). Nowa wersja mo-

```
06 USER S0          ( SZCZYT PUSTEGO STOSU DANYCH)
08 USER R0          ( SZCZYT PUSTEGO STOSU POWROTOW)
0A USER TIB         ( BUFOR WEJSCIOWY TERMINALA)
0C USER WIDTH       ( MAKSYMALNA ILOSC ZNAKOW W NAZWIE)
0E USER WARNING     ( STERUJE REAKCJA NA BLEDY)
10 USER FENCE       ( GRANICA ZAPOMINANIA DEFINICJI PRZY UZYCIU FORGET)
12 USER DP          ( WSKAZNIK SLOWNIKA)
14 USER VOC-LINK    ( WSKAZNIK DO PFA+6 OSTATNIO ZDEFINIOWANEGO SLOWNIKA)
16 USER BLK         ( NUMER INTERPRETOWANEGO BLOKU LUB 0 GDY TIB)
18 USER IN          ( WSKAZNIK ZNAKOW BUFORA WEJSCIOWEGO)
1A USER OUT         ( POZYCJA KURSORA NA EKRANIE)
1C USER SCR         ( NUMER EDYTOWANEGO KADRU)
1E USER OFFSET      ( 0 -GDY WYBR. DRIVE 0, IL. SEK. NA DYSKU - GDY DR.1)
20 USER CONTEXT     ( ADRES PFA+4 SLOWNIKA PRZESZUKIWANEGO JAKO )
                    (          PIERWSZY PRZY KOMPILOWANIU SLOWA )
22 USER CURRENT     ( ADR. PFA+4 SLOWN. DO KT. DOLACZA SIE DEF.)
24 USER STATE       ( 00 - STAN EGZEKUCJI, C0 - KOMPILACJI)
26 USER BASE        ( PODSTAWA SYSTEMU LICZBOWEGO)
28 USER DPL         ( POZYCJA KROPKI DZIESIETNEJ)
2A USER FLD         ( SZEROKOSC POLA WYSWIETLANEJ LICZBY )
2C USER CSP         ( MIEJSCE NA ZAPAMIETYWANIE WSKAZNIKA STOSU)
2E USER R#          ( POZYCJA KURSORA EDYTORA)
30 USER HLD         ( WSKAZNIK OSTATNIEGO ZNAKU W BUFORZE PAD)
```

Wydruk 3

```
: 1+ 1 + ;                                  ( ZWIEKSZ O 1 WARTOSC NA STOSIE)
: 2+ 2 + ;                                  ( ZWIEKSZ O 2 WARTOSC NA STOSIE)
: HERE DP @ ;                       ( PODAJE NASTEPNY WOLNY ADRES W SLOWNIKU)
: ALLOT DP +! ;                             ( PRZESUN WSKAZNIK SLOWNIKA)
: , HERE ! 2 ALLOT ;                ( WPISZ DO SLOWNIKA WARTOSC ZE STOSU)
: C, HERE C! 1 ALLOT ;                      ( WPISZ DO SLOWNIKA BAJT ZE STOSU)
: - MINUS + ;                       ( ODEJMIJ WARTOSC OSTATNIA OD PRZEDOSTATNIEJ)
: = - 0= ;                          ( WARTOSC LOGICZNA 'PRZEDOSTATNIA=OSTATNIA')
: < - 0< ;                          ( WARTOSC LOGICZNA 'PRZEDOSTATNIA<OSTATNIA')
: U< 2DUP XOR 0< IF DROP 0< 0= ELSE - 0< ENDIF ;  ( NIEJEDNOSC BEZ ZNAKU)
: ROT >R SWAP R> SWAP ;             ( ROTUJ PRZEDPRZEDOSTATNIA NA WIERZCH)
: SPACE BL EMIT ;                           ( WYSWIETL SPACJE NA TERMINALU)
: -DUP DUP IF DUP ENDIF ;                   ( DUPLIKUJ WARTOSC NIEZEROWA)



: TRAVERSE                                  ( PRZESUN SIE POPRZEZ POLE NAZWY)
                                    ( ADR-2 KIER-1 --- ADR-1     , -1(--, +1--))
   SWAP BEGIN OVER + 7F OVER C@ < UNTIL SWAP DROP ;
: LATEST CURRENT @ @ ;                      ( NFA OSTATNIEGO SLOWA W CURRENT)
: LFA 4 - ;                                 ( ZAMIEN PFA SLOWA NA LFA)
: CFA 2 - ;                                 ( ZAMIEN PFA SLOWA NA CFA)
: NFA 5 - -1 TRAVERSE ;                     ( ZAMIEN PFA SLOWA NA NFA)
: PFA 1 TRAVERSE 5 + ;                      ( ZAMIEN NFA SLOWA NA PFA)
: !CSP SP@ CSP ! ;                  ( ZAPAMIETAJ POLOZENIE SZCZYTU STOSU W 'CSP')
: ?ERROR                    ( FLAG-2, NUMER BLEDU-1, OSTRZEZENIE JESLI FLAG-1)
        SWAP IF ERROR ELSE DROP ENDIF ;
: ?COMP STATE @ 0= 11 ?ERROR ;              ( BLAD JESLI NIE STAN KOMPILACJI)
: ?EXEC STATE @ 12 ?ERROR ;                 ( BLAD JESLI NIE STAN EGZEKUCJI)
: ?PAIRS - 13 ?ERROR ;  ( BLAD JESLI WARTOSCI NA STOSIE NIE SA IDENTYCZNE)
: ?CSP SP@ CSP @ - 14 ?ERROR ;      ( SPRAWDZ POLOZENIE WSKAZNIKA STOSU)
: ?LOADING                  ( BLAD JESLI INTERPRETACJA NIE Z BUFORA DYSKOWEGO)
        BLK @ 0= 16 ?ERROR ;
: COMPILE       ( KOMPILUJ ADRES, KTORY JEST NASTEPNY W RAMACH DEFINICJI)
        ?COMP R> DUP 2+ >R @ , ;
: [ 0 STATE ! ; IMMEDIATE                   ( ZATRZYMAJ KOMPILACJE)
: ] C0 STATE ! ;                            ( WEJDZ W STAN KOMPILACJI)
: SMUDGE LATEST 20 TOGGLE ;  ( ZMIEN BIT SMUDGE W OSTATNIM NAGLOWKU SLOWA)
: HEX 10 BASE ! ;                           ( PRZYJMIJ PODSTAWE 16D DLA WE-WY)
: DECIMAL 0A BASE ! ;                       ( PRZYJMIJ PODSTAWE 10D DLA WE-WY)
: (;CODE)   ( ZAPISZ POLE KODU WSKAZNIKIEM DO DALSZEGO OBSZARU DEFINICJI)
        R> LATEST PFA CFA ! ;
: ;CODE ?CSP COMPILE (;CODE)                ( KONCZ NOWE SLOWO DEFINIUJACE)
        [COMPILE] [ SMUDGE NOOP ( ASSEMBLER ) ; IMMEDIATE
: (BUILDS 0 CONSTANT ;                      ( UTWORZ NAGLOWEK SLOWA)
: DOES) ( WPISZ AD. WYS. POZIOMU DO PFA, DO CFA - AD. KODU 'DOES)'-DODOE)
    R> LATEST PFA ! ;CODE ( ... KOD PROCEDURY DOCOL ...) C;
: COUNT DUP 1+ SWAP C@ ;            ( USTAL ADRES-2, LICZBE ZNAKOW-1 TEKSTU)
: TYPE                      ( WYPROWADZ TEKST OD ADRESU-2, LICZBA ZNAKOW-1)
      -DUP IF OVER + SWAP DO I C@ EMIT LOOP ELSE DROP ENDIF ;
: -TRAILING             ( USTAL LICZBE ZNAKOW PO ODRZUCENIU KONCOWYCH SPACJI)
      DUP 0 DO OVER OVER + 1 - C@ BL - IF LEAVE ELSE 1 - ENDIF LOOP ;
: (.")          ( WYPROWADZ UMIESZCZONY DALEJ TEKST, DOSTOSUJ SLAD POWROTU)
    R COUNT DUP 1+ R> + >R TYPE ;
: ."    22 STATE @          ( KOMPILUJ LUB WYPROWADZ TEKST POMIEDZY ." A ")
    IF COMPILE (.") WORD HERE C@ 1+ ALLOT
    ELSE WORD HERE COUNT TYPE ENDIF ;       IMMEDIATE
: EXPECT                        ( ADRES BUFORA NA TEKST-2, LIMIT ZNAKOW -1)
    OVER + OVER
      DO KEY DUP 0E +ORIGIN @ =
        IF DROP DUP I = DUP R> 2 - + >R IF 07 ELSE 08 ENDIF
        ELSE DUP 0D = IF LEAVE DROP BL 0 ELSE DUP ENDIF I C! 0 I 1+ !
        ENDIF EMIT
      LOOP DROP ;
```

```
: QUERY TIB @ 50 EXPECT 0 IN ! ;  ( POBIERZ TEKST Z KONSOLI DO BUFORA TIB)
8081 ( ZAWART. POLA NAZWY ZLOZ. Z JEDNEGO ZNAKU '00' ) HERE ( ADR. POLA N.)
: X      BLK @  (  NULL KONCZACY W BUFORZE TEXT - POWROT NA WYZSZY POZIOM)
 IF ( DYSK.) 1 BLK +! 0 IN ! BLK @ B/SCR 1 - AND 0= IF ?EXEC R) DROP ENDIF
 ELSE ( TIB) R) DROP ENDIF ; ! ( ZAMIEN NAZWE X NA '00' ) IMMEDIATE
: FILL                          ( WYPELNIJ PAMIEC OD ADR-3, ILOSCIA-2 BAJTOW-1)
    SWAP )R OVER C! DUP 1( R) 1 - CMOVE ;
: ERASE 0 FILL ;                ( WYPELNIJ PAMIEC ZERAMI OD ADR-2, ILOSC -1)
: BLANKS BL FILL ;              ( WYPELNIJ PAMIEC SPACJAMI OD ADR-2, ILOSC-1)
: HOLD -1 HLD +! HLD @ C! ;                  ( SCHOWAJ ZNAK W BUFORZE PAD)
: PAD HERE 44 + ;                ( PAD ZNAJDUJE SIE 68D BAJTOW POWYZEJ HERE)
: WORD  ( WPROW. TEKST WEJSC.  AZ DO OGRANICZNIKA, UMIESC TEKST W BUFORZE)
     BLK @ IF BLK @ BLOCK ELSE TIB @ ENDIF          ( I PRZESLIJ DO 'HERE')
     IN @ + SWAP                                  ( ADRES-2, OGRANICZNIK-1)
     ENCLOSE          ( ADRES-4, POCZATEK-3, KONIEC-2, CALKOWITA-ILOSC-1)
     HERE 22 BLANKS                  ( WYCZYSC BUFOR WPISUJAC 34D SPACJE)
     IN +!                             ( PRZESKOCZ PRZEZ LANCUCH W TIB)
     OVER - )R                                      ( CHRON ILOSC ZNAKOW)
     R HERE C!                        ( PRZED TEKSTEM ZAPISZ ILOSC ZNAKOW)
     + HERE 1+
     R) CMOVE ;           ( PRZEPISZ LANCUCH ZNAKOW Z BUFORA TIB DO HERE+1)
: (NUMBER)                                      ( D1 ADR1 ---> D2 ADR2)
                ( DODAJ DO PODWOJNEJ LICZBY D1 LICZBE ZAPISANA W POSTACI)
   ( ZNAKOW ASCII POD ADR1, WYNIK PODWOJNA LICZBA D2, ADR2=ADR1+IL.ZNAKOW)
 BEGIN 1+ DUP )R C@ BASE @ DIGIT
 WHILE SWAP BASE @ UM* DROP ROT BASE @ UM* D+ DPL @ 1+ IF 1 DPL +! ENDIF R)
 REPEAT R) ;
: NUMBER ( ZAMIEN ZN. ASCII POD ADR NA LICZBE PODWOJNA D; JESLI NAPOTKANO)
                                          ( . 10 POZYCJE KROPKI ZAPAMIETAJ W DPL )
 0 0 ROT DUP 1+ C@ 2D = DUP )R + -1
 BEGIN DPL ! (NUMBER) DUP C@ BL -
  WHILE DUP C@ 2E - 0 ?ERROR 0 REPEAT
  DROP R) IF DMINUS ENDIF ;
: -FIND                                         ( ZNAJDZ SLOWO W SLOWNIKU)
                    ( PO WYKONANIU: PFA 3, BAJT DLUGOSCI-2,1 ALBO TYLKO 0)
   BL WORD HERE CONTEXT @ @ (FIND)
   DUP 0= IF DROP HERE LATEST (FIND) ENDIF ;
: (ABORT) ABORT ;                    ( ABORT ZMIENIANY PRZEZ UZYTKOWNIKA)
: ERROR (  W ZALEZNOSCI OD WARNING:)
     ( -1=ABORT, 0=BRAK DYSKOW-NUMER BLEDU, 1= NA DYSKI-KOMUN. O BLEDZIE)
  WARNING @ 0<
  IF (ABORT) ENDIF HERE COUNT TYPE ." ? "
  MESSAGE SP! BLK @ -DUP IF IN @ SWAP ENDIF QUIT ;
: ID.                          ( WYPROWADZ NAZWE NA PODSTAWIE ADRESU NFA)
    PAD 020 5F FILL DUP PFA LFA OVER
    PAD SWAP CMOVE PAD COUNT 01F AND TYPE SPACE ;



: CREATE  ( UTWORZ NAGLOWEK Z USTAWIONYM BITEM SMUDGE, WYSLIJ OSTRZEZENIE)
                        ( JESLI NOWA NAZWA JUZ WCZESNIEJ BYLA ZDEFINIOWANA)
    TIB HERE 0A0 + U< 2 ?ERROR -FIND
    IF ( OSTRZEZENIE) DROP NFA ID. 4 MESSAGE SPACE ENDIF
    HERE DUP C@ WIDTH @ MIN 1+ ALLOT
    DUP 0A0 TOGGLE HERE 1 - 80 TOGGLE    ( USTAW BITY BAJTOW GRANICZNYCH)
    LATEST , CURRENT @ !
    HERE 2+ , ;
: [COMPILE]                        ( WYMUSZA KOMPILACJE NATYCHMIASTOWEGO SLOWA)
    -FIND 0= 0 ?ERROR DROP CFA , ; IMMEDIATE
: LITERAL          ( JESLI STAN KOMPILACJI, KOMPILUJE LIT I LICZBE ZE STOSU)
    STATE @ IF COMPILE LIT , ENDIF ; IMMEDIATE
: DLITERAL     ( JESLI STAN KOMPILACJI, KOMPILUJE PODWOJNA LICZBE ZE STOSU)
    STATE @ IF SWAP [COMPILE] LITERAL [COMPILE] LITERAL ENDIF ; IMMEDIATE
: ?STACK  ( SPRAWDZ, CZY WSKAZNIK STOSU NIE WYSZEDL POZA DOPUSZCZ. ZAKRES)
SP@ S0 @ SWAP U< 1 ?ERROR SP@ HERE 80 + U< 7 ?ERROR ;
: INTERPRET     ( INTERPRETUJ LUB KOMPILUJ SLOWA ZRODL. TEKSTU WEJSCIOWEGO)
    BEGIN -FIND
        IF ( ZNALEZIONO) STATE @ (
            IF ( KOMPILUJ) , ELSE ( WYKONAJ) EXECUTE ENDIF ?STACK
        ELSE ( NIE ZNALEZIONO) HERE NUMBER DPL @ 1+
            IF ( BYLA KROPKA) [COMPILE] DLITERAL
            ELSE ( NIE BYLO KROPKI) DROP [COMPILE] LITERAL
            ENDIF ?STACK
        ENDIF
    AGAIN ;
: IMMEDIATE        ( ZMIEN BIT POPRZEDZANIA (PRECEDENCE) OSTATNIEGO SLOWA)
    LATEST 40 TOGGLE ;
: VOCABULARY                       ( UTWORZ SLOWO BEDACE NOWYM SLOWNIKIEM)
    <BUILDS A081 , CURRENT @ CFA , HERE VOC-LINK @ , VOC-LINK !
    DOES> 2+ CONTEXT ! ;
VOCABULARY FORTH IMMEDIATE                                ( SLOWNIK FORTH )
: DEFINITIONS                    ( ZE SLOWNIKA CONTEXT UCZYN TAKZE CURRENT)
    CONTEXT @ CURRENT ! ;
: (                ( PRZESKOCZ TEXT WEJSCIOWY AZ DO PRAWEGO NAWIASU OKACZNIE)
    29 ( PRAWY NAWIAS) WORD ; IMMEDIATE
: QUIT                                    ( RESTART, INTERPRETUJ Z TERMINALA)
   0 BLK ! [COMPILE] [
   BEGIN RP! CR QUERY INTERPRET STATE @ 0= IF ." OK" ENDIF AGAIN ;
: ABORT SP! HEX ?STACK CR ." FIG FORTH V1.1"          ( "CORACY" RESTART)
    [COMPILE] FORTH DEFINITIONS QUIT ;
```

Wydruk 4

że być pozbawiona pola nazwy i pola łącznika w definicjach katalogowych, gdyż przy realizacji konkretnego zadania nie są one potrzebne. Daje to możliwość uzyskania krótkiego kodu (często krótszego od napisanego w asemblerze), przy jednoczesnym szybkim pisaniu i uruchamianiu programu. Proces generowania nowych wersji FORTHA nazywany jest **metakompilacją.**

ZBIGNIEW POJMAŃSKI

Stosowanie kodu maszynowego w komputerach osobistych

Popularne komputery osobiste zamiast programu MONITOR wyposażone są zazwyczaj w interpreter języka BASIC umieszczony w pamięci stałej. Ma to swoje zalety, jeśli przyjąć, że użytkownikiem będzie tzw. odbiorca masowy, który rzadko potrzebuje bardziej wyrafinowanych narzędzi. Niestety, mikroprocesory 8-bitowe mają zbyt małą moc obliczeniową, aby można było nawet w popularnych zastosowaniach ograniczyć się do stosowania wyłącznie języków wysokiego poziomu. Zachodzi więc konieczność wykorzystania, przynajmniej w niektórych podprogramach, kodu maszynowego.

W interpreterach języka BASIC stosowanych w komputerach osobistych zazwyczaj występuje funkcja **USR adr,** która pozwala wywołać podprogram w języku maszynowym, rozpoczynający się od adresu **adr.** Procedura w języku maszynowym powinna kończyć się rozkazem **RETURN,** co pozwala na ponowne przejęcie sterowania przez interpreter języka BASIC. Jeśli zachodzi potrzeba przekazania danej, to powinna ona zostać umieszczona w parze rejestrów BC (dla mikrokomputerów wykorzystujących Z80 lub 8080). Jeżeli więc napiszemy **LET A=USR adr,** to zmienna A będzie zawierała wartość jaka była w BC w chwili powrotu z podprogramu. Gdy danych jest więcej — procedura w języku maszynowym może zapisać je w komórkach pamięci o określonym adresie. Potem wystarczy wykonać **PEEK adr,** aby wykorzystać je w programie napisanym w języku BASIC. Analogicznie przez **POKE** można przekazać dane dla procedury w języku maszynowym. Gdy nie zachodzi potrzeba przekazania danych z podprogramu, najczęściej stosuje się zapis: **RANDOMIZE USR adr.**

Pozostaje jeszcze kwestia zapisu programów w języku maszynowym. Tradycyjną i zarazem najwygodniejszą metodą jest wykorzystanie do tego celu pogramów EDITOR i ASSEMBLER [1]. Pierwszy pozwala na wprowadzenie programu w postaci symbolicznej. Powstaje w ten sposób tzw. wersja źródłowa programu (ang. source), która jest znacznie bardziej przejrzysta dla człowieka niż zestaw cyferek. Za pomocą programu ASSEMBLER z wersji źródłowej naszego programu otrzymujemy tzw. wersję wynikową, czyli kod, który po zapisaniu do pamięci może być interpretowany przez procesor.

W niektórych mikrokomputerach programy EDITOR i ASSEMBLER zawarte są w pamięci stałej. W przypadku komputerów osobistych trzeba je zazwyczaj ładować do pamięci z taśmy magnetofonowej lub z dysku.

Co zrobić, jeśli nie dysponujemy programami EDITOR i ASSEMBLER? Praktycznie nie warto zabierać się za pisanie własnych programów w języku maszynowym. Wprawdzie można pobawić się wyszukując w książce kod dla każdej instrukcji, ale ma to sens tylko w przypadku bardzo krótkich procedur.

Tylko niektóre komputery osobiste mają możliwość zapamiętania programu w kodzie maszynowym poprzez zapisanie go na taśmie (lub dysku). Przykładowo — dla ZX81 trzeba zastosować pewien trik; kod maszynowy umieszcza się w pierwszej linii programu za instrukcją REM — w ten sposób nietrudno określić adres początku, a kod razem z resztą programu można bez problemu zapisać na kasecie magnetofonowej.

---

[1]) Słowo ASSEMBLER używane jest również dla określenia języka maszynowego

Przy publikowaniu programów dla mikrokomputerów bardzo często fragmenty pisane w języku maszynowym zamieszcza się pod postacią tablicy kodów. Drukowanie wersji źródłowej zajmuje znacznie więcej miejsca i dodatkowo ogranicza krąg odbiorców — do tych, którzy dysponują programem ASSEMBLER. Wprowadzanie (sprawdzonych już przez kogoś) kodów jest zapewne bardziej żmudne niż wprowadzanie wersji źródłowej — sumarycznie zajmuje jednak mniej czasu. Kody zazwyczaj drukuje się w liniach zawierających: adres, 16 cyfr szesnastkowych i sumę kontrolną. Suma kontrolna pozwala na wykrycie błędów popełnionych przez wprowadzającego. Dla ułatwienia wprowadzania stosuje się specjalne programy ładujące kody szesnastkowe we wskazane miejsce pamięci (ang. hex loader).

Począwszy od tego wydania mikroKLANU publikować będziemy co jakiś czas programy dla użytkowników ZX81. Jeśli będą one zawierać fragmenty napisane w języku maszynowym, proponujemy do ich wprowadzania wykorzystywać następujący program ładujący:

```
6900 REM ZAPIS
7000 CLS
7010 PRINT "WPISYWANIE CZY SPRAWDZANIE? W/S"
7020 INPUT M$
7030 PRINT "ADRES POCZATKU"
7040 INPUT S
7050 PRINT "ADRES KONCA"
7060 INPUT F
7070 IF M$="S" THEN GOTO 8000
7080 FOR N=S TO F STEP 8
7090 LET T=0
7100 PRINT N;"—";
7110 INPUT A$
7120 PRINT A$;"=";
7130 INPUT TOT
7140 PRINT TOT
7145 LET Z=0
7150 FOR K=1 TO LEN A$ STEP 2
7160 LET C=(CODE A$(K)—28)*16+CODE A$(K+1)—28
7170 LET T=T+C
7180 POKE N+Z,C
7190 LET Z=Z+1
7200 NEXT K
7210 IF TOT=T THEN GOTO 7240
7220 SCROLL
7225 PRINT "BLAD — JESZCZE RAZ"
7230 GOTO 7090
7240 NEXT N
7250 STOP
7900 REM SPRAWDZANIE
8000 FOR N=S TO F STEP 8
8005 SCROLL
8010 PRINT N;"—";
8020 FOR B=0 TO 7
8030 LET C=PEEK (N+B)
8040 LET F$=CHR$ (INT (C/16+28))
8050 LET S$=CHR$((C—((CODE F$—28)*16))+28
8055 PRINT F$;S$;
8060 NEXT B
8070 PRINT
8080 NEXT N
```

Zanim użyty zostanie program ładujący należy napisać: 1 REM i tyle znaków, ile bajtów liczy program. Znaki te mogą być najzupełniej dowolne — ważna jest ich liczba.

Zaprezentowany program ładujący można również wykorzystać i dla innych mikrokomputerów (np. ZX SPECTRUM). Należy go jednak odpowiednio przystosować (ZX81 nie wykorzystuje kodów ASCII).

AJP

prowadzi:
Andrzej J. Piotrowski
tel. dom. 48-22-85
tel. służb. 43-66-11 w. 98

Wielu czytelników mikroKLANU stawia dopiero swoje pierwsze mikroinformatyczne kroki. Nie jesteśmy w stanie przy obecnej objętości mikroKLANU zamieszczać materiałów na poziomie zupełnie elementarnym. Wspomagając tych co stawiają krok drugi lub trzeci publikujemy między innymi proste programy w języku BASIC, wychodząc z założenia, że najlepiej uczyć się na przykładach.

Zamiana liczb arabskich na rzymskie

```
 5 REM ZAMIANA LICZB ARABSKICH NA RZYMSKIE
 6 REM RZYMIANIE NIE MIELI OZNACZENIA NA
 7 REM LICZBY MNIEJSZE NIZ 1 ORAZ NA
 8 REM WIEKSZE NIZ 4999.
 10 DIM A(13),R(13)
 20 FOR I=1 TO 13
 30 READ A(I)
 40 NEXT I
 50 DATA 1000,900,500,400,100,90,50,40,10,9,5,4,1
 60 PRINT "LICZBA ARABSKA (1-4999)?"
 61 INPUT N
 70 IF N<1 OR N>4999 THEN GOTO 60
 80 PRINT N;"=";
 90 FOR I=1 TO 13
 100 R(I)=0
 110 M=N-A(I)
 120 IF M<0 THEN GOTO 160
 130 R(I)=R(I)+1
 140 N=M
 150 GOTO 110
 160 NEXT I
 165 IF R(1)=0 THEN GOTO 200
 170 FOR I=1 TO R(1)
 180 PRINT "M";
 190 NEXT I
 200 IF R(2)=1 THEN PRINT "CM";
 210 IF R(3)=1 THEN PRINT "D";
 220 IF R(4)=1 THEN PRINT "CD";
 225 IF R(5)=0 THEN GOTO 260
 230 FOR I=1 TO R(5)
 240 PRINT "C";
 250 NEXT I
 260 IF R(6)=1 THEN PRINT "XC";
 270 IF R(7)=1 THEN PRINT "L";
 280 IF R(8)=1 THEN PRINT "XL";
 285 IF R(9)=0 THEN GOTO 320
 290 FOR I=1 TO R(9)
 300 PRINT "X";
 310 NEXT I
 320 IF R(10)=1 THEN PRINT "IX";
 330 IF R(11)=1 THEN PRINT "V";
 340 IF R(12)=1 THEN PRINT "IV";
 345 IF R(13)=0 THEN GOTO 380
 350 FOR I=1 TO R(13)
 360 PRINT "I";
 370 NEXT I
 380 PRINT
 390 PRINT "NASTEPNA LICZBA (0-KONIEC)?"
 391 INPUT N
 400 IF N>0 THEN GOTO 70
 410 END
READY.
    RUN
LICZBA ARABSKA (1-4999)?
? 1984
 1984 =MCMLXXXIV
NASTEPNA LICZBA (0-KONIEC)?
? 2865
 2865 =MMDCCCLXV
NASTEPNA LICZBA (0-KONIEC)?

? 0
READY.
```

J.T.

W mikroKLANIE 6 pod artykułem Władysława Minora nt. mikrokomputera NEPTUN 184 (INFORMATYKA nr 6/84, s. 14) zamieściliśmy następującą adnotację: „Artykuł ten został opublikowany w porozumieniu z producentem i przez niego sfinansowany". Wywołała ona nieporozumienia, autor zaś poczuł się urażony posądzeniem, iż występuje jako rzecznik producenta. Wyjaśniamy więc, że — zgodnie z wcześniejszymi zapowiedziami — staramy się uzyskać od wytwórców sprzętu mikrokomputerowego dotacje finansowe związane z artykułami nt. ich produktów, przy czym „cenzury producenta" nie uznajemy. Pracownia Elektroniki Medycznej, firmująca NEPTUNA, jako pierwsza przyjęła, że wsparcie INFORMATYKI ma szersze uzasadnienie; zgodziła się więc na wykupienie kolumn pisma przeznaczonych na artykuł Władysława Minora. Sam autor w ogóle o tej transakcji nie wiedział — tekst nie został w jej wyniku naruszony. (Red.)

dokończenie ze str. 12

System MYCIN „zna" aktualny stan swojej wiedzy i sposób jej reprezentowania dzięki wprowadzeniu tzw. profilów (ang. profiles) oraz metareguł.

Profile są dołączone do każdego z dwudziestu czterech predykatów i dają systemowi możliwość stwierdzenia, które atrybuty są zarządzane przez dane predykaty. Każdy atrybut może występować w wielu przesłankach różnych reguł. Profile służą do szybkiego przybliżonego oszacowania, które z przesłanek należy w danej chwili rozważać. Oszacowanie to wykazuje, czy reguła może być zastosowana, odrzucona czy też należy decyzję o jej zastosowaniu odłożyć na później.

Metareguły narzucają systemowi strategię przeszukiwania bazy danych. Na przykład — metareguła definiująca kolejność czynności, jakie należy podjąć przy określaniu terapii, może być następująca:

if określana jest terapia
then rozważaj reguły, które umożliwiają:
1) zdobycie danych o stanie pacjenta
2) wykrycie rodzaju bakterii, które mogły spowodować zakażenie
3) identyfikację najbardziej prawdopodobnej bakterii
4) znalezienie najużyteczniejszych leków
5) wybór możliwie najmniejszej ilości odpowiedzi.

Zauważmy, że metareguły mają charakter heurystyczny.

* * *

System MYCIN-TEIRESIAS został zaprogramowany w języku LISP na maszynie PDP-10. Pięciu specjalistów niezależnych od twórców programów zbadało działanie systemu. W 72% przypadków wyniki uzyskane przez system były poprawne. W pozostałych przypadkach specjaliści nie zgadzali się w diagnozie między sobą.

Wiele spośród istniejących systemów ekspertowych wykazuje dużą wagę programów spełniających podobną rolę jak TEIRESIAS. Przykładowo — system PUFF wspomagający diagnozę chorób płucnych wymagał najwyżej pięćdziesięciu godzin pracy ekspertów nad uzyskaniem przez system odpowiedniego stanu wiedzy. Uruchomienie PUFFA zajęło dziesięć osobo-tygodni pracy.

Warto wspomnieć w tym miejscu o niedociągnięciach systemu MYCIN:

- Najsłabszym jego punktem jest przetwarzanie języka naturalnego (program BAOBAB).
- Jeśli system MYCIN zawodzi z powodu konieczności wprowadzenia lub zmodyfikowania kilku reguł jednocześnie, to zawodzi również w tym przypadku program TEIRESIAS.
- Przy wprowadzaniu nowej reguły TEIRESIAS zakłada z góry, iż jest ona wątpliwa, a aktualny stan wiedzy systemu właściwy. Jeśli więc zachodzi przypadek odwrotny — system może mieć trudności.

Pomimo tych niedoskonałości, MYCIN jest bardzo dobrym i użytecznym systemem ekspertowym. Zarządza on specjalistyczną wiedzą w sposób efektywny, stale ją poprawiając i rozszerzając — dzięki zdobywaniu nowych doświadczeń. Co więcej — jest systemem łatwym do implementacji, a jego zachowanie jest proste do zrozumienia, weryfikacji i zaakceptowania przez specjalistów. Wszystkie te cechy powodują, iż MYCIN można zaliczyć do najbardziej udanych systemów ekspertowych.

LITERATURA

[1] Bonnet A.: Analyse de textes au moyen d'une grammaire sementique et de schémas. Applications á la comprehension de résumés medicaux en langage naturel, Thése d'Etat, Paris, 6, 1980
[2] Davis R.: Interactive transfer of expertise: acquisition of new inference rules. MIT, 11, pp. 1—36, 1979
[3] Doyle J.: TMS: a true maintenance system. A. I., 12, pp. 231—272, 1979
[4] Friedman L.: Extended plausible inference. IJCAI, 7, pp. 487—495, 1981
[5] Shortliffe E.: Computer-based Medical Consultations: MYCIN. Elsevier, USA, 1976
[6] Swartout W. R.: Explaining and justifying expert consulting programs, IJCAI, 7. pp. 815—823, 1981.

Opracował
ANDRZEJ SZAŁAS

na podstawie artykułu J. L. Lauriere'a „Expert systems" Technology and Science of Informatics, vol. 1, no. 1, 1983

Zasady prenumeraty

(dokończenie ze str. 6)

PRENUMERATA ze zleceniem wysyłki za granicę jest dwukrotnie droższa.

PRENUMERATA ULGOWA. Uprawnieni są do niej: indywidualni członkowie Stowarzyszeń Naukowo-Technicznych, studenci oraz uczniowie szkół zawodowych. Warunkiem prenumeraty ulgowej jest poświadczenie blankietu przekazu (na odwrocie środkowego odcinka) pieczęcią Koła SNT NOT, wyższej uczelni lub szkoły.

UWAGA!

1. Dotychczasowi prenumeratorzy otrzymają przesyłkę pocztową „Informator tytułowy i cennik czasopism technicznych" oraz blankiet wpłaty-zamówienia z Zakładu Kolportażu.
2. Instytucje zainteresowane prenumeratą naszego czasopisma mogą otrzymać cennik i blankiet-zamówienie w Zakładzie Kolportażu oraz w:
   — Biurach Wydawniczych:
   90-020 Łódź, Pl. Komuny Paryskiej 5a
   50-019 Wrocław, ul. Świerczewskiego 74
   — Oddziale Wydawnictwa:
   40-014 Katowice, ul. Dąbrowskiego 23
   — Oddziałach Wojewódzkich NOT
   — Zarządach Głównych Stowarzyszeń Naukowo-Technicznych
   — Redakcjach czasopism branżowych.
3. Powyższe warunki prenumeraty dotyczą czasopism kolportowanych przez Wydawnictwo NOT SIGMA.

Dodatkowych informacji o prenumeracie udziela Zakład Kolportażu, tel. 40-00-21 wewn. 293, 299 lub 40-35-89.

EGZEMPLARZE ARCHIWALNE czasopism wydawanych przez Wydawnictwo NOT SIGMA (od 1977 roku) można zamawiać w Zakładzie Kolportażu, 00-950 Warszawa ul. Mazowiecka 12 (tel. 26-80-16) lub nabywać w Klubie Prasy Technicznej, 00-950 Warszawa, ul. Mazowiecka 12 (tel. 27-43-65).

Ośrodek Przetwarzania Danych
ZAKŁADÓW WŁÓKIEN CHEMICZNYCH
»STILON«
w Gorzowie Wlkp., ul. Fr. Walczaka 25
tel. 72-23-42, teleks 044624 stil.

ma do upłynnienia
sprawny
używany sprzęt informatyczny:

- zestawy numeryczne Addo X 6 szt.
- dziurkarki kart Soemtron 415. 3 szt.
- sprawdzarki kart Soemtron 425 2 szt.
- części zamienne do wymienionych urządzeń

EO/991/K/84

I/C. Buforowanie i blokowanie

Niniejszy zestaw pytań [1]) nie stanowi jakiegokolwiek standardu egzaminacyjnego i dlatego stosowanie go do sprawdzania wiedzy innej osoby — poza samym sobą — stanowi świadome pogwałcenie intencji ACM i zastrzeżonych praw autorskich tego stowarzyszenia.

PYTANIA

21. Które z poniższych określeń nie nadaje się do opisania funkcji bufora występującego między procesem wysyłającym a odbierającym?

a) bufor wyrównuje różnice w szybkościach obu procesów
b) bufor uniezależnia w procesie odbierającym szybkość przetwarzania otrzymywanego komunikatu od szybkości procesu wysyłającego
c) bufor umożliwia, by proces wysyłania w pewnych sytuacjach generował komunikat z szybkością większą niż ta, z jaką proces odbierający jest w stanie go przetworzyć
d) bufor może ograniczać — przez swą skończoną pojemność — prędkość, z jaką działa proces wysyłający

22. Rozważmy sytuację, w której ma nastąpić zapis na taśmę magnetyczną: 1) dane są generowane przez jednostkę centralną znacznie szybciej niż mogą być zapisywane na taśmie, 2) jednostka taśmowa jest połączona z jednostką centralną, 3) na obszar buforowy przeznaczono ograniczoną, wydzieloną na stałe część pamięci operacyjnej. Ile powinno być prawidłowo buforów?

a) 4
b) tyle buforów, ile jednostek taśmowych
c) 2
d) tyle buforów o pojemności 1K, ile mieści się w wyznaczonej części pamięci operacyjnej

23. Koncepcja podwójnego buforowania pozwala na:

a) tworzenie większych bloków wejściowych i wyjściowych
b) zapisanie w bloku fizycznym dwa razy więcej rekordów logicznych
c) jednoczesne wykonywanie (ang. overlapping) operacji we-wy w jednym buforze — a przetwarzania w innym
d) dynamiczny przydział buforów

24. Które z poniższych stwierdzeń jest prawdziwe dla buforowania cyrkulacyjnego (ang. circular buffering), przy założeniu, że bufor ma pojemność co najmniej dwóch rekordów:

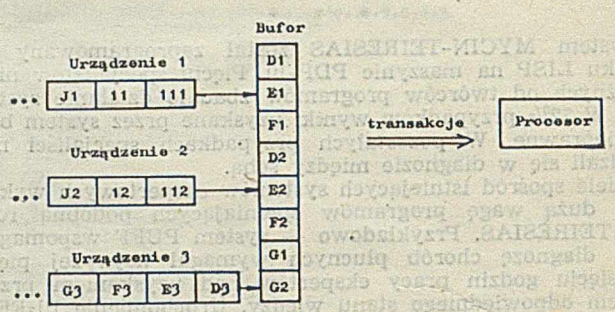
a) proces wysyłający nie może umieścić danych w buforze, dopóki proces odbierający nie opróżni tego buforu całkowicie
b) proces wysyłający nie może umieścić danych w buforze w tym samym czasie, kiedy proces odbierający pobiera dane z tego buforu
c) proces wysyłający po napotkaniu najwyższego adresu fizycznego w danym buforze musi zostać zatrzymany
d) procesy wysyłający i odbierający dzielą się tym samym buforem w kolejności rosnących adresów miejsca

25. Rozpatrzmy sytuację, w której procesy A i B są niezależnie zaplanowane:

Która z poniższych przykładowych kolejności przetwarzania — przy niezależnym zaplanowaniu obu procesów — prowadzi w konsekwencji do zakleszczenia:

a) A1, A2, B1, A3, A4, B2, B3, B4
b) A1, A2, A3, A4, B1, B2, B3, B4
c) B1, B2, A1, B3, B4, A2, A3, A4
d) A1, B1, A2, B2, A3, B3, A4, B4

26. Rozważmy sytuację, w której: 1) trzy urządzenia wejściowe dysponują wspólnym buforem, 2) procesor nie może rozpocząć przetwarzania kolejnej transakcji, a więc zwolnić obszaru buforu, dopóki wszystkie rekordy danego typu (np. wszystkie A_i dla $i = 1, 2, 3$) nie będą dostępne w buforze, 3) wszystkie trzy urządzenia wejściowe działają niezależnie od siebie, 4) bufor może pomieścić tylko osiem rekordów



Wskaż najbardziej właściwy opis dla sytuacji pokazanej powyżej:

a) system jest w stanie oczekiwania (ang. wait)
b) system jest w stanie zajętości (ang. busy)
c) system jest w stanie zakleszczenia (ang. deadlocked)
d) system jest w stanie pracy nakładkowej (ang. overlapped)

27. Rozważmy sytuację, w której 400 tys. niezblokowanych rekordów 100-znakowych przechowywanych jest na ośmiu krążkach taśmy magnetycznej o następującej charakterystyce: długość krążka 730 m (2400 stóp), gęstość zapisu 63 znaki/mm (1600 znaków/cal), przerwa międzyblokowa 12,7 mm (0,5 cala). Załóżmy, że rekordy są zblokowane po 20. Ile potrzeba krążków taśmy magnetycznej, aby można było zapisać cały ten zbiór?

a) 1; b) 2; c) 3; d) 4

28. Rozważmy proces przetwarzania, którego szybkość jest limitowana szybkością czytania ze zbioru taśmowego bloków o 2400 znakach. Każdy blok zawiera trzy rekordy logiczne po 800 znaków. Urządzenie taśmowe ma następującą charakterystykę: gęstość 63 znaki/mm (1600 znaków/cal), czas zatrzymania taśmy 0,5 s, czas rozbiegu 0,5 s, szybkość czytania 1270 mm/s (50 cali/s). Załóżmy, że dokonujemy zmiany blokowania na dziewięć rekordów logicznych w jednym bloku. Ile wówczas wyniesie przypuszczalny współczynnik przyspieszenia czasu przetwarzania?

a) 3; b) 6; c) 9; d) 27

29. Mamy dysk magnetyczny, w którym każda ścieżka podzielona jest na 12 sektorów, po 120 znaków każdy. Załóżmy, że mamy zapisać na tym dysku zbiór zawierający rekordy logiczne 90-znakowe. Który z poniż-

---

[1]) Communications of the ACM., Vol. 19 (no. 5, May 1981) p. 231. Stowarzyszenie ACM zezwoliło na przedruk Self-Assessment Procedure I do celów niekomercyjnych (Copyright by Association for Computing Machinery)

szych współczynników blokowania spowoduje optymalne wykorzystanie obszaru dysku?

a) 1; b) 2; c) 3; d) 4

30. Zbiór główny o pojemności 500 tys. rekordów logicznych pięćdziesięcioznakowych zapisany jest na dysku w blokach po 20 rekordów. Zbiór ten jest wykorzystywany cztery razy dziennie do przetwarzania zbioru transakcyjnego, przy czym za każdym przebiegiem uzyskuje się dostęp do 5% rekordów zbioru głównego. Aktywność rekordów w zbiorze rozkłada się równomiernie w całym zbiorze. Przy którym z niżej podanych sposobów przetwarzania zostanie zminimalizowany czas pojedynczego takiego przebiegu?

a) przetwarzanie transakcji w kolejności ich pojawiania się, z przypadkowym dostępem do zbioru głównego
b) tworzenie zbioru transakcyjnego, sortowanie transakcji i sekwencyjne przeszukiwanie zbioru głównego
c) tworzenie zbioru transakcyjnego i indeksowo-sekwencyjne przeszukiwanie zbioru głównego
d) tworzenie zbioru transakcyjnego i losowe przeszukiwanie zbioru głównego.

ROZWIĄZANIA

21 b, 22 c, 23 c, 24 d, 25 d, 26 c, 27 b, 28 a, 29 d, 30 b

KOMENTARZE

21. Brinch Hansen P.: Podstawy systemów operacyjnych. WNT. pp. 89—93

22. Gear C. W.: Computer Organization and Programming. McGraw-Hill, II wyd., pp. 260—261

23. Shaw A. C.: Projektowanie logiczne systemów operacyjnych. WNT, 1980, s. 45—50

24. Shaw A. C.: ibid. s. 45—50

25. Madnick S., Donovan J.: Systemy operacyjne. PUN, 1983 Brinch Hansen, P.: ibid. pp. 122—130

26. Madnick S., Donovan J.: ibid

27. Bohl M.: Information Processing. Science Research Associates, Dokładne wyliczenie: (20 [rek./blok] × 100 [zn./rek] × (1/1600) [cal/zn.] × 0,5 [cal/blok]) × 400 000 [rek./plik] × (1/20) [blok rek.] × × (1/12) [stopa/cal] × (1/2400) [taśma/stopa] = 1,22 [taśma/plik] ≈ ≈ 2 krążki taśmy

28. Drummond M.: Evaluation and Measurement Techniques for Digital Computer Systems. Prentice-Hall, pp. 27—30
Czas odczytu jednego rekordu wynosi: 800 [zn./rek.] × (1/1600) [cal/zn]. × (1/50) [s/cal] = 0,01 [s/rek.]
Czas start/stopu dla jednego bloku wynosi 0,5 [s/blok] + 0,5 [s/ /blok] = 1 [s/blok]
Czas odczytu n rekordów:
— dla współczynnika zblokowania równego 3 wynosi:
(0,01 [s/rek.] + 1 [s/blok] × (1/3) [blok rek.]) × n [rek.] = 0,34 × × n [s]
— dla współczynnika zblokowania równego 9 wynosi:
(0,01 [s/rek.) + 1 [s/blok] × (1/9) [blok rek.]) × n [rek.] = 0,12 × × n [s] a więc około trzykrotnie mniej

29. Eckhouse R.: Systemy minikomputerowe. Organizacja i programowanie. WNT
Najlepszy układ: całkowita liczba bloków przypadających na całkowitą liczbę sektorów — tj. 360 znaków na 3 sektory przy 4 rekordach w bloku

30. Współczynnik aktywności 5% przy współczynniku blokowania równym 20 oznacza wartość oczekiwaną dostępu rzędu jednego rekordu na blok. Ponieważ wszystkie bloki są równodostępne — najlepszym podejściem byłoby posortowanie wsadu i przetworzenie go w jednym przebiegu.

Szanowna Redakcjo!

Zdecydujcie się, czyje interesy chcecie reprezentować: wyspecjalizowanych teoretyków programowania czy „szarych" szeregowych analityków, projektantów, programistów. Z jednej strony jesteście zatroskani w swoich felietonach (p. artykuł Janusza Gwiazdy w nr 1 z 1984 r.) stanem informatyki w Polsce, z drugiej strony — nie robicie prawie nic, aby ten stan poprawić. Jedynym jasnym punktem w INFORMATYCE jest mikroKLAN, niestety jednak — jak zwykle — postawiony przez Was na głowie.

Powiedzmy sobie szczerze: środowisko informatyków, jak chyba żadne inne, jest uzależnione od „środowiska" użytkowników. Ileż powstało „półkowych" systemów z powodu złego zdefiniowania celu przez użytkownika, ile zakupiono miernej jakości sprzętu z powodu „ignorancji" użytkownika itd.

Zdecydujcie się: albo jesteście pismem inżynierskim i wtedy nie wylewajcie „krokodylich" łez nad stanem informatyki w Polsce, albo zajmujcie się informatyką przede wszystkim z punktu widzenia użytkownika.

Banalne jest stwierdzenie, ale niestety prawdziwe, że im bardziej informatyka zrozumiała jest dla użytkownika, tym większe są szanse na zrobienie czegoś sensownego i z pożytkiem dla społeczeństwa. Opinia o informatyce w Polsce wyrabia się nie na podstawie tego, że mgr X wymyślił algorytm skracający czas wykonania obliczeń o 0,3 s (90% objętości pisma), ale na podstawie działania systemów kulejących od początku ich tworzenia. INFORMATYKA powinna się zająć nie dokształcaniem programistów, a „wychowaniem" użytkownika — powtarzając, aż do znudzenia w wielu wariantach, tak banalne prawdy, jak:
— informatyka nie jest celem samym w sobie, ale jednym z narzędzi w przedsiębiorstwie
— może to być „impreza" bardzo droga
— dobra organizacja i sprawny obieg informacji w firmie jest podstawą właściwego użycia technik komputerowych
— „personal" potrafi wprowadzić taki sam bałagan w firmie, jak duży komputer
— komputer nie zastąpi człowieka w myśleniu
— głupie dane wejściowe prowadzą do jeszcze głupszych danych wyjściowych.

Chciałbym powtórzyć jeszcze raz: stan informatyki w Polsce zależy tylko i wyłącznie od aktualnych i potencjalnych użytkowników, a nie od programistów. Nie dajmy się „zrobić na szaro", jak to przytrafiło się technikom i inżynierom.

WŁODZIMIERZ STANECKI

Warszawa, 29 maja 1984

Od Redakcji

W istocie rzeczy musimy się zgodzić z Włodzimierzem Stanecki. Ma on rację, wykazując nam brak konsekwencji. Ale główny postulat, skierowany do nas, trafia pod zły adres. Wydajemy pismo dla informatyków, bywa im ono czasem potrzebne — trudno by nam było po prostu zmienić czytelnika. Faktem jest jednak, że nie ma w Polsce czasopisma dla użytkowników. Stąd różne „dodatkowe" tematy na naszych łamach.

Wychodzimy z założenia, że większy wpływ na użytkowników mają pisma branżowe, gdzie omawia się lokalne zastosowania informatyki, oraz pisma o szerszej tematyce (Horyzonty Techniki, Przegląd Techniczny, Problemy, Forum), w których informatyka może się pojawić jako problem ogólny. Pisma te jednak — zapewne w efekcie niedoboru kadry — nie są dobrym propagatorem informatyki. A my nie jesteśmy w stanie przejąć tej roli.

Być może powinno powstać nowe, wysokonakładowe pismo o informatyce dla niefachowców. Ale o tym na pewno nie my zadecydujemy.

Staramy się podejmować problemy związane ze społecznym kontekstem informatyki, choćby po to, by sami informatycy potrafili szerzej patrzeć na swą dziedzinę. „Krokodylich łez" może w istocie wylaliśmy za dużo. Próbujemy już pisać bardziej rzeczowo.

Na marginesie: nieprawdą jest jakoby pismo zawierało w przewadze banalne doniesienia racjonalizatorskie. Wystarczy uważniejsza lektura... Tę inwektywę zrzucamy na karb polemicznego zacięcia.

I jeszcze prośba. Chętnie przeczytalibyśmy wnikliwsze uwagi na temat oczekiwanego mikroKLANU „na nogach".

Z.G.

Międzynarodowe Targi Książki

W dniach 23—28 maja 1984 odbyły się w Warszawie po raz 29 Międzynarodowe Targi Książki. Była to jedna z niewielu możliwości przejrzenia aktualnie wydanych książek z dziedziny informatyki. Książki obejmujące szeroko pojętą informatykę cieszą się w świecie dużym popytem i — osiągając znaczne nakłady — prezentują różnorodną gamę tematyki oraz różne stopnie opisu zagadnień.

W Targach wzięło udział ponad 200 oficyn wydawniczych z 21 krajów. Nas mogły zainteresować przede wszystkim wydawnictwa: Academic Press, Prentice-Hall i Pergamon Press z Anglii, Addison Wesley z Holandii, McGraw-Hill z USA (filia w RFN) oraz Springer Verlag z Berlina Zachodniego. Znane też wydawnictwo John Wiley and Sons uczestniczyło tylko pośrednio w ramach innych wydawnictw.

Największą grupę książek z naszej dziedziny stanowiły, jak zawsze, książki opisujące języki programowania. Liczba tytułów sugeruje, że najpopularniejszymi językami są nadal BASIC, PASCAL i COBOL. Należy przy tym zaznaczyć, że — w odróżnieniu od dawnych lat — nowe opisy tych języków podkreślają w tytułach i spisach treści prezentację zaawansowanych, strukturalnych metod programowania (np. [1]). Liczne były też pozycje o języku ADA, przy czym w większości zawierały one wstęp do języka lub omówienie podręcznika języka, często z włączeniem w treść znanych jego fragmentów in extenso. Na uwagę — moim zdaniem — zasługuje praca Habermanna i Perry'ego [2]. Opisy pozostałych najważniejszych języków, takich jak FORTRAN, FORTRAN 77 i PL/1 występowały sporadycznie. Pojedyncze były też opisy języka FORTH, MODULA, MODULA-2 czy SMALLTALK-80 [3] oraz NPL [4].

Oczywiście, większość książek tej grupy zawierała omówienie wersji języka dla konkretnego typu komputera, głównie VAX-11 oraz komputerów osobistych: IBM PC, APPLE, COMMODORE, ZX SPECTRUM, BBC SYNERTEK itp. Oprócz opisów języków, dla komputerów osobistych, bogaty był też zestaw pozycji omawiających w sposób mniej lub bardziej skomplikowany systemy operacyjne CP/M i UNIX [5], VISICALC, edytory i programy redagowania tekstów typu WORDSTAR [6], grafikę mikrokomputerową, a także — zawierających zbiory programów statystycznych i innych przydatnych dla posiadacza komputera osobistego, nie wyłączając gier komputerowych i opracowań dla dzieci (z nauką programowania w języku LOGO). O metodach walki konkurencyjnej na tym rynku niech świadczy, fakt, że w okładce książki o języku PASCAL znajdowała się minidyskietka z kompilatorem tego języka.

Pozostawmy jednak ten rynek nastawiony na zdobycie masowego użytkownika komputera. W zakresie tematyki profesjonalnej, liczba pozycji nie odbiegała już od przeciętnej. Przede wszystkim pojawiło się kilka książek dotyczących piątej generacji komputerów. Książki te obejmowały szeroką tematykę, począwszy od omówienia technik VLSI [7], poprzez zbiory artykułów z konferencji, aż do esejów na tematy postępu i celowości tych prac [8]. Zaznaczyć też należy prezentację nowego pisma „New Generation Computing", wydawanego przez Springer Verlag i OMSHA Ltd. (z Japonii), którego redaktorem naczelnym jest T. Moto-Oka.

Sprzęt komputerowy był opisywany głównie z punktu widzenia architektury, organizacji i projektowania 8- i 16-bitowych systemów mikrokomputerowych. Wiele pozycji, powielając opisy katalogowe, wyjaśniało od podstaw konstrukcję takich systemów. Do ciekawszych prac z tego zakresu zaliczyłbym opis rodziny mikroprocesorów 8086/8088 [9]. Interesującą pozycją jest też publikacja o nowoczesnych rozwiązaniach bardzo szybkich pamięci [10].

Książki na temat oprogramowania dotyczyły głównie zastosowań struktur danych [11] oraz budowy różnego rodzaju baz danych. Interesujące były też pozycje dotyczące metod konstruowania programów grafiki komputerowej. Znacznie zmniejszyła się natomiast podaż książek o tematyce dotyczącej budowy oraz projektowania systemów operacyjnych, a te, które były prezentowane, nie stanowiły rewelacji. Na przykład, książka znanych autorów Petersona i Silberschaza [12], traktując — według reklamówki — o drugiej generacji systemów operacyjnych, jest klasyczną prezentacją wielu dawno już znanych i opisanych metod. Również niewiele ciekawego można było znaleźć na temat systemów czasu rzeczywistego (np. [13]) czy też ich zastosowań w przemyśle. Dla projektantów kompilatorów godna polecenia mogłaby być jedynie książka Waite'a i Goos'a [14].

Pojedyncze pozycje omawiały zagadnienia budowy sieci komputerowych oraz zasad konstruowania i weryfikacji protokołów. W temacie metod optymalizacji, badania wydajności i strojenia systemów najciekawszą pozycją była praca Ferrariego, Serraziego i Zeignera [15], będąca rozwinięciem poprzednich opracowań. Pojawiły się też pierwsze książki dotyczące systemów ekspertowych [16].

Na zakończenie tego przeglądu zwracam uwagę na serię Lecture Notes in Computer Science (LCNS), która osiągając już 164 tom zyskała sobie uznanie pośród pracowników naukowych informatyki [17].

W sumie, dla biblioteki specjalizującej się w dziedzinie informatyki można było na targach wybrać około sto interesujących pozycji.

Jeszcze parę słów o cenach książek. Ceny te są wysokie, przeciętnie od 20 do 50 dolarów, zależnie od rodzaju okładki, metody druku (wiele pozycji było drukowanych bezpośrednio z egzemplarza przygotowanego przez autora — z wykorzystaniem redakcji komputerowej) oraz rangi publikacji. Niektóre woluminy, jak opracowania typu „State of the Art" z Pergamon Press sięgnęły ceny ponad 400 dolarów. Oczywiście bez względu na rodzaj przyjętego kursu wymiany walut, książki te są raczej niedostępne dla indywidualnego odbiorcy. Z drugiej strony — także w warunkach zachodnich — książka naukowa jest droga i, co ciekawsze, w wielu krajach nawet droższa niż to wynika z relacji walut. Przykładem jest tutaj Dania, gdzie wartość książki w koronach duńskich jest znacznie większa od równowartości jej ceny w dolarach. Jest to tym bardziej zadziwiające, że ze względu na niewielką grupę informatyków w tym kraju, wydawanie książek w języku duńskim jest po prostu nieopłacalne. Tak więc nawet biblioteki uniwersyteckie nie dysponują tam pełnym zestawem nowości.

Na koniec — pytanie: gdzie w Polsce możemy zdobyć do przeczytania interesujące nas pozycje. Po Targach książki te zostały rozdysponowane do bibliotek specjalistycznych PAN, instytutów naukowych oraz szkół wyższych i tam należy ich poszukiwać. W sytuacji braku dewiz na zakupy za granicą, eksponaty z Targów stają się szczególnie cennym źródłem nowości literatury informatycznej.

WACŁAW ISZKOWSKI

DANE BIBLIOGRAFICZNE OMÓWIONYCH KSIĄŻEK

[1] Koffman E. B., Friedman F. L.: Problem Solving in Structured Basic-Plus and VAX-11 Basic. Addison Wesley, 1984

[2] Habermann A. N., Perry D. E.: Ada for Experienced Programmers. Addison Wesley, 1984

[3] Goldberg A., Robson D., Krasner G.: Smalltalk-80 (seria 4 książek). Addison Wesley, 1983—1984

[4] Truitt T. D., Mindlin S. B.: An Introduction to Nonprocedural Languages using NPL. McGraw-Hill, 1984

[5] Sobell M. G.: A Practical Guide to the UNIX System. Addison Wesley, 1984; McGilton H., Morgan: Introducing to the UNIX system, McGraw-Hill, 1984

[6] Knuth D. E.: Computers and Typesetting, Vol. 1. TEX Vol. 2 The TEX book. Addison Wesley, 1984 (na targach nie było egzemplarza tej bardzo interesującej pozycji)

[7] Randell B., Treleaven P.: VISI Architecture. Prentice-Hall, 1983 (50 dol.)

[8] Feingenbaum E. A., McCorduck P.: Fifth Generation, Artificial Intelligence and Japan's Challenge to the World. Addison Wesley, 1983

[9] Gibsen G. A.: Microcomputer Systems (the 8086/8088 Family — Architecture, Programming and Design). Prentice Hall, 1984 (44,50 dol.)
[10] Pohm A. V., Agrawal O. P.: High Speed Memory Systems. Prentice-Hall, 1984
[11] Lewis T. G.: Software Engineering, Analysis and Verification. Prentice-Hall, 1984
[12] Peterson J., Silberschatz A.: Operating Systems Concepts. Addison Wesley, 1983
[13] Glass R. L.: Real-Time Software. Prentice-Hall, 1984 (33,30 dol.)
[14] Waite, Goos: Compiler Construction. Springer Verlag, 1984 (69 DM)
[15] Ferrari D., Serrazi G., Zeigner A.: Measurement and Tuning of Computer Systems. Prentice Hall, 1983 (47,95 dol.)
[16] Hayes-Roth F., Watermann D., Lenat D.: Building Expert Systems. Addison Wesley, 1984
[17] Lecture Notes in Computer Science. Springer Verlag:
a) Ed. Nakajima, Yuasa: The IOTA Programming System, A Modular Programming Environment, Vol. 160. 1983
b) Goos, Wulf, Evens Jr, Butler: DIANA, An Intermediate Language for Ada, Revised Version, Vol. 161, 1983
c) Ed. van Hulzen: Computer Algebra. EUROCAL'83 Conference, London 1983 Vol. 162, 1983
d) Ed. Clarke, Kozen: Logics in Program. Workshop Pittsburgh 1983, Vol. 164 1984.

Rada ds. Zastosowań Środków Techniki Obliczeniowej

System bazy danych dla przedsiębiorstw

Po ogólnej prezentacji Rady ds. Zastosowań Środków Techniki Obliczeniowej (INFORMATYKA nr 2'1983) oraz jej bieżących spraw organizacyjnych (6, 1984) pora na prezentację realizacji konkretnych tematów. Jak wspomniano w pierwszym artykule, polska część Rady odpowiada za wykonanie ok. dwudziestu pozycji Jednolitego Planu Współpracy. Wśród nich znajduje się temat: Modułowy system informatyczny przedsiębiorstwa. Warto podkreślić, że realizacja tego przedsięwzięcia polega na uzyskaniu rzeczywistego pakietu programów, a nie założeń lub zaleceń.

Zapewne dla większości Czytelników hasła: MOSIP i RODAN — nie są obce. Przypomnijmy jednak, że RODAN to system zarządzania bazą danych, wywodzący się z Centrum Projektowania i Zastosowań Informatyki, a MOSIP jest akronimem Modułowego Systemu Informatycznego Przedsiębiorstwa, powstałego i nadal rozwijanego w Instytucie Organizacji Przemysłu Maszynowego (IOPM).

IOPM, realizując temat w ramach prac RZ ŚTO, wykorzystuje standardowe algorytmy systemu MOSIP w systemie bazy danych dla przedsiębiorstw, użytkujących SBD/TP (System Bazy Danych z Teleprzetwarzaniem) RODAN. Obecny stan zaawansowania prac pozwala już na prowadzenie wdrożeń pilotowych systemu RODAN-MOSIP-STANDARD. Na rok 1985 zaplanowano jego odbiór międzynarodowy, według procedury Komisji Międzyrządowej ds. ETO.

Główny cel systemu to zinformatyzowanie przedsiębiorstw przemysłowych, których zastosowania wymagają tworzenia i utrzymywania złożonych struktur danych oraz charakteryzują się szybko zmieniającymi się wymaganiami w zakresie funkcji użytkowych.

System RODAN-MOSIP-STANDARD działa w oparciu o dwa bloki programowe, zawierające następujące procedury:

- systemu narzędziowego SBD/TP RODAN
- organizowania i obsługi bazy danych systemu oraz obsługi użytkownika.

Informacje zasilające system bazy danych mogą pochodzić z różnych źródeł, w tym również z innych systemów informatycznych eksploatowanych w przedsiębiorstwie oraz z tradycyjnych dokumentów źródłowych. Informacje wyjściowe są realizowane przez procedury mechanizmu obsługi użytkownika. Mechanizm ten zawiera obecnie zestaw wydruków standardowych i jest przystosowany do łatwej rozbudowy. W krótkim czasie może być on — na przykład — uzupełniony o zestaw wydruków lub komunikatów wyświetlanych na monitorze ekranowym, zgodnie ze specyficznymi wymaganiami użytkownika.

System może być eksploatowany na komputerach Jednolitego Systemu: R-32, R-40, R-50 i R-60 oraz na IBM 360/370. Komputer musi być wyposażony w jeden z następujących systemów operacyjnych: OS/MVT, OS/MFT, OS/VS1 lub OS/VS2 oraz w system narzędziowy SBD/TP RODAN.

Koszty Eksploatacji systemu RODAN-MOSIP-STANDARD są uzależnione od liczby realizowanych funkcji użytkowych oraz struktury systemu. Na życzenie użytkownika, IOPM wykonuje badania eksploatacyjne tego systemu w zakresie określonym przez użytkownika i na danych dostarczonych przez niego lub będących w posiadaniu IOPM.

Czas wdrożenia systemu od chwili dostarczenia danych przez użytkownika wynosi ok. sześciu miesięcy, przy czym IOPM deklaruje daleko idącą pomoc w procesie przygotowania danych.

Należy podkreślić, że RODAN-MOSIP-STANDARD jest systemem otwartym tak w obszarze realizowanych funkcji użytkowych, jak i co do możliwości rozbudowy strukturalnej. W obszarze realizowanych funkcji użytkowych system ma charakter modułowy, może być więc wdrażany etapami — w zależności od potrzeb użytkownika. Istnieje możliwość zwiększenia zakresu funkcji użytkowych — docelowo do poziomu kompleksowego. W ramach rozbudowy strukturalnej — istniejące mechanizmy SBD/TP RODAN umożliwiają pracę systemu w trybie wielozadaniowym oraz z zastosowaniem teleprzetwarzania i języka bezpośredniego użytkownika.

Podstawowe zalety systemu to łatwość modyfikacji rozbudowy oraz krótki czas wdrożenia. Zakłada się, że jego głównymi użytkownikami będą jedno- lub wielozakładowe przedsiębiorstwa przemysłowe o produkcji seryjnej, zamierzające rozszerzać zastosowanie informatyki.

Oprac.
MKS

EO/432/K/84

ZAKŁAD ELEKTRONICZNEJ TECHNIKI OBLICZENIOWEJ
ul. Ofiar Oświęcimskich 7/13
50-069 Wrocław

oferuje do sprzedania:

- części zamienne do maszyn:
  ARITMA
  Facit
  TELETYPE
  ODRA 1304 i 1305
  Soemtron
- półprzewodniki:
  diody
  układy scalone
  tranzystory
  rezystory

Informacji udziela:
Dział Zaopatrzenia ZETO
tel. 44-27-62
teleks: 0712533 ZETO PL

EO/937/K/84

Komputerowa szansa niepełnosprawnych

W ostatnich latach technika rehabilitacji inwalidów poczyniła wielkie postępy, także dzięki zastosowaniom mikroelektroniki. Coraz więcej pieniędzy przeznacza się na badania i zakup urządzeń ułatwiających niepełnosprawnym normalne życie w społeczeństwie. Niedawno (INFORMATYKA nr 4, 1984, III okł.) przedstawiliśmy inicjatywę rządu brytyjskiego przyjścia z pomocą upośledzonym, dziś prezentujemy opracowanie obszernego artykułu z tygodnika „Die Zeit" nr 47, 1983. Autorzy przedstawiają obecny stan i perspektywy rehabilitacji osób niepełnosprawnych fizycznie, pokazując jednocześnie jak wiele jest jeszcze do zrobienia w tej dziedzinie. A mogłoby się wydawać, że na obecnym poziomie technologii konstrukcja aparatów umożliwiających inwalidom normalne funkcjonowanie nie powinna nastręczać specjalnych trudności.

23-letnia pacjentka jest od pięciu lat sparaliżowana od pasa w dół. Dr Petrofsky z Uniwersytetu Wrighta w Dayton postawił ją znów na nogi. Do mięśni jej nóg i pleców przymocowano 36 elektrod, ukrytych pod ubraniem. Mały komputer, umocowany do paska, koordynuje bodźce elektryczne, które zmuszają mięśnie do kontrakcji i w ten sposób regulują stanie i chód. Czujniki — w tym również takie, jakich używa się w rakietach międzykontynentalnych — informują komputer o pozycji nóg i tułowia. Komputer z kolei przekazuje swoje informacje do elektrod i w ten sposób steruje procesem poruszania się krok po kroku.

Jeszcze przed kilku laty pacjentka ta w najlepszym przypadku mogłaby poruszać się za pomocą ogromnej aparatury, bardziej podobna do robota niż człowieka. Dr Petrofsky ma nadzieję, że w przyszłości można będzie wszczepić jej pod skórę czujniki, wyposażone w mikroprocesory, podobne do rozruszników serca. Maszynę do chodzenia, tzn. mały komputer, będzie wtedy mogła ona nosić bez zwracania uwagi otoczenia — jako coś w rodzaju damskiej torby.

Jest to jeden z wielu przykładów, które wkrótce zrewolucjonizują rehabilitację najcięższych przypadków kalectwa. Kilka kroków zrobionych przez wspomnianą pacjentkę to w istocie ogromny skok — skomputeryzowana maszyna do chodzenia to osiągnięcie, które można równać z wynalezieniem okularów lub ze skonstruowaniem przed 500 laty pierwszego fotela na kółkach.

Współczesny stan techniki umożliwia realizację pomysłów, które jeszcze przed kilku laty wydawały się utopią. W przyszłości, dzięki protezom opartym na mikroelektronice, sparaliżowani mają znów chodzić, niewidomi — widzieć, głusi — słyszeć.

Według szacunków Światowej Organizacji Zdrowia, na świecie jest kilka milionów ludzi o znacznym stopniu kalectwa. Wiele milionów innych — mimo nawet potencjalnych zdolności — prowadzi nędzną egzystencję. Mikroelektronika, telekomunikacja, robotyka, elektroniczna synteza i analiza mowy, osiągnięcia w dziedzinie sztucznej inteligencji, teoria sytemów, automatyczne odczytywanie rysunków i znaków, nowe materiały dające się łączyć z żywym organizmem, banki danych, wideodyski, otwierają ludziom niepełnosprawnym możliwość usamodzielnienia się. Czołowi specjaliści od zagadnień rehabilitacji spodziewają się, że w najbliższych dziesięciu latach nastąpią na tym polu wielkie zmiany.

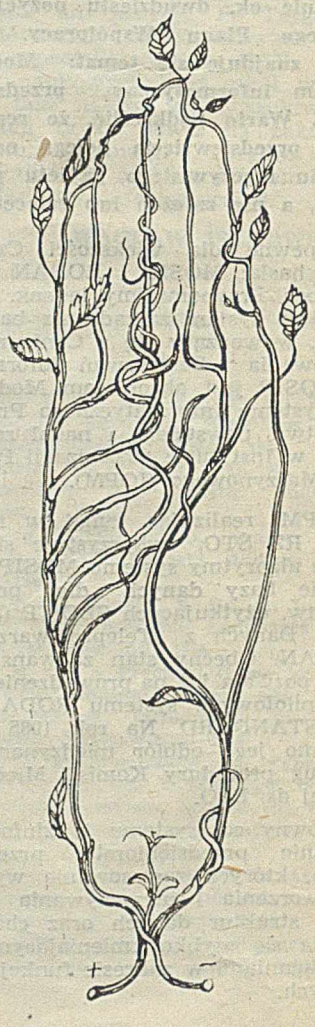
Rehabilitacja uchodziła dotąd za pasierba medycyny. Obecnie problematyką tą interesują się najsławniejsze uniwersytety. Uniwersytet Johna Hopkinsa w Baltimore ogłosił ogólnokrajowy konkurs na zastosowanie komputerów w terapii ludzi niepełnosprawnych. W Stanford trwają prace konstrukcyjne nad inteligentnym robotem obsługującym chorych z zaburzeniami narządów ruchu. W San Francisco i w Wiedniu wszczepia się głuchym „rozruszniki słuchu". W Tokio prowadzi się w jednej z klinik doświadczenia z robotami. W MIT i Chicago realizuje się ambitne projekty dotyczące neuroprotez. Również przemysł zauważył niezwykły rozwój w technice rehabilitacji. Wiele małych firm opracowuje programy komputerowe przystosowane do potrzeb ludzi niepełnosprawnych, zgłasza nowe patenty. Wiele z tych firm zostało założonych właśnie przez osoby niepełnosprawne.

Byłoby jednak nierozsądne oczekiwać rozwiązania od razu wszystkich problemów ludzi upośledzonych. Doświadczeń z codziennych kontaktów z inwalidami nie da się niczym zastąpić. W żmudnej, drobiazgowej, systematycznej pracy specjaliści od gimnastyki rehabilitacyjnej, lekarze i wszyscy inni terapeuci osiągają wyniki nie zawsze dostatecznie doceniane przez techników. Bez zainteresowania i zachęty, które są częścią tradycyjnej terapii, pomoc techniczna niczego nie da. Maszyny i aparaty nie mogą zastąpić kontaktu z innymi ludźmi. Ich zastosowanie będzie miało trwały pozytywny skutek tylko wtedy, kiedy będzie pomagało osobom niepełnosprawnym odnaleźć własne miejsce w społeczeństwie.

W badaniach naukowych i technice można wyróżnić trzy kierunki prowadzące do tego celu:

- Przez wszczepianie „inteligentych protez" przywraca się utracone funkcje ciała. Analogicznie do rozrusznika serca, można mówić o rozruszniku mięśni, słuchu i. o protezach optycznych.
- Pomocy ludziom niepełnosprawnym mają udzielać „inteligentne aparaty". Niemi mogą mówić za pomocą maszyn, niewidomym urządzenia odczytują gazety i książki. Roboty obsługują paralityków, wózki inwalidzkie same odnajdują drogę.
- Automatyzuje się trening i ćwiczenia (gimnastykę rehabilitacyjną, terapię mowy, trening poznawczy w przypadku chorych z uszkodzeniami czaszki i mózgu) — tak, że mogą one trwać dłużej niż zwyczajowe pół godziny dziennie.

Jeden z najambitniejszych projektów mających na celu stawianie na nogi ludzi sparaliżowanych, a chorym z bezwładem rąk ponowne umożliwienie chwytania przedmiotów — jest realizowany od około dwudziestu lat w Lublianie (Jugosławia). Stosując skoordynowane pobudzanie mięśni elektrycznością, próbuje się tu u sparaliżowanych pacjentów zastąpić utracone funkcje ciała. Podobnie jak w przypadku zakłóceń rytmu serca, można pobudzać elektrycznie mięsień sercowy za pomocą wszczepionej elektrody i w ten sposób zmuszać serce we właściwym momencie do kontrakcji, tak przy zastosowaniu elektrycznej stymu-



Rys. Jacek Gawłowski

lacji funkcji mięśnie połączone z kośćcem pobudzane są przez elektrody w celu wywołania ruchu.

Jednak jest to o wiele bardziej skomplikowane niż w przypadku rozrusznika serca. Podczas gdy cały mięsień sercowy może pobudzać jedna elektroda, to nawet w prostych czynnościach chwytania uczestniczą dziesiątki grup mięśni, które muszą być stymulowane w ścisłej koordynacji czasowej i z odpowiednio dobraną siłą. Problem ten może rozwiązać tylko komputer, który jednak musi być tak mały, żeby można było wszczepić go jako „rozrusznik mięśni". Nad tym problemem pracują dziś różne grupy naukowców w Cleveland, Los Angeles, Dayton, Chicago, Londynie, Lublianie i Wiedniu.

Minie zapewne jeszcze wiele lat, zanim metoda ta wyjdzie ze stadium eksperymentów. Mimo to można już odnotować pewne sukcesy. W stosowanym obecnie „rozruszniku mięśni" drugiej generacji aparat stymulujący i mikroprocesory przymocowane są do wózka inwalidzkiego. Ruchy pacjenta są jeszcze bardzo niezręczne, ale bądź co bądź może wziąć do bezwładnej dotąd ręki szklankę, podnieść ją do ust i pić. Kierownicy programów stosowania neuroprotez nie chcą jeszcze wzbudzać w pacjentach zbyt wielkich nadziei, trzeba pokonać tu wiele istotnych trudności. Obecnie rozwiązania są jeszcze bardzo niedoskonałe i można je stosować tylko u chorych z bardzo silną motywacją.

W 1972 r. w Los Angeles trzem głuchym pacjentom po raz pierwszy wszczepiono elektryczne aparaty stymulujące, które przekształcały zewnętrzne bodźce dźwiękowe w elektryczne pobudzanie nerwów słuchowych. Te pierwsze stymulatory miały jedną elektrodę pobudzającą i mogły przekazywać do mózgu tylko niektóre bodźce dźwiękowe. Chociaż prymitywna, metoda ta przyniosła korzyści większości pacjentów. Dziś około 460 ludzi nosi „rozrusznik słuchu" przekazujący za pomocą bodźców elektrycznych informacje akustyczne bezpośrednio do mózgu. Nowe aparaty mają do 16 elektrod. Niektórym z głuchych pacjentów udało się nawet za ich pomocą nauczyć się obcego języka, co wskazuje na dobry zmysł słuchu. Ale i tu potrzeba cierpliwości ponieważ metody te wymagają jeszcze wielu lat prób.

Od dawna prowadzi się również badania nad protezami wzroku, pobudzającymi korę mózgową. Wielkie sumy przeznacza się również na perspektywiczne badania nad kostkami białkowymi, które można by wszczepiać w mózg. Ale już dziś istnieje wiele nowych środków, które w zasadniczy sposób mogą zmienić życie inwalidów. W 1978 r. XEROX wprowadził na rynek czytnik dla niewidomych, który do dzisiaj znalazł się już w wielu bibliotekach. Za pośrednictwem syntetyzatora nowy aparat czyta niewidomym normalnie drukowane książki lub drukuje teksty pismem Braille'a.

Dla osób z zaburzeniami mowy dostępne są „protezy mowy". Jeden z popularnych systemów, MINSPEAK, polega na tym, że niemowa ma przed sobą tablicę z 60 polami, oznaczonymi symbolami i uruchamianymi przez dotyk. Przez wskazanie czterech czy pięciu symboli powstaje logiczny związek, przekształcany przez mikrokomputer wyposażony w syntetyzator mowy w pełne, mówione zdania. Jednakże powstaje tu trudność polegająca na tym, że normalną rozmową prowadzi się z szybkością ok. 180 słów na minutę, natomiast przy użyciu opisanego aparatu można osiągnąć najwyżej 15 słów na minutę. Obserwacje dowiodły, że na skutek tej różnicy w szybkości porozumiewania się zostają naruszone podstawowe reguły konwersacji. Pojawia się zniecierpliwienie i człowiek zdrowy szybko kończy rozmowę. Niektóre ośrodki badawcze pracują więc nad dalszym udoskonaleniem tej metody.

W Waszyngtonie niewidomy makler giełdowy wykonuje swój zawód tak, że wielu klientów w ogóle nie wie o jego kalectwie. Używa on mówiącego mikrokomputera. Informacja jest wyświetlana na ekranie i odczytywana głośno, przy czym może się to odbywać z różnymi prędkościami. Ponieważ na Wall Streat decyduje szybkość, tempo odczytywania można zwiększyć do 540 słów na minutę. Za pośrednictwem komputera makler jest bezpośrednio połączony z giełdą, kupuje, sprzedaje i zgłasza opcje. Specjalne rozwiązania programowe pozwalają mu opuszczać lub odczytywać tylko określone fragmenty tabeli kursów. Są już niewidomi sędziowie, adwokaci, nauczyciele i sekretarki, którzy także posługują się tym systemem.

Komputer osobisty to klucz do niezależności dla ludzi niepełnosprawnych. W Instytucie Rehabilitacji w Chicago — na przykład — pracuje pacjentka z tetraplegią, która może poruszać tylko ustami i głową, a mimo to jako telefonistka przyjmuje telefony i dokonuje połączeń, w przerwach zaś — przepisuje teksty. Posługuje się ona podobnym do słomki ustnikiem: wciągając lub wydmuchując powietrze może — za pomocą odpowiedniego programu — zdumiewająco szybko wywołać litery lub numery telefoniczne oraz pisać na maszynie listy.

Założone przed ośmiu laty w Illinois przedsiębiorstwo szkoli w zawodzie programisty ludzi o dużym stopniu inwalidztwa — sparaliżowanych, chorych na stwardnienie rozsiane, pacjentów z zanikiem mięśni — tak by mogli pracować jako programiści komputerów. Firma ta wstawia im do domu terminal oraz modem. Szkolenie trwa pół roku, po którym inwalida otrzymuje umowę o pracę na rok i pracuje dla jednej z 40 firm, z którymi firma zawarła porozumienie.

Na początku lat siedemdziesiątych uczeni z kliniki uniwersyteckiej w Heildelbergu (RFN) jako pierwsi zaczęli próby ze stosowaniem robotów przemysłowych w rehabilitacji. Po kilku latach zaprzestano jednak prób, ponieważ stosunek kosztów do uzyskanych efektów był nie do zaakceptowania przez pacjentów. Przed podobnymi trudnościami stoją uczeni uniwersytetów w Stanford oraz Johna Hopkinsa, którzy, obok Japończyków, realizują największe projekty w tej dziedzinie. Próbuje się ograniczyć do minimum czynności umożliwiające posługiwanie się robotem. Roboty firmy UNIMATION — PUMA są już tak dobrze wyposażone, że reagują na polecenia mówione. Słownictwo takiego robota obejmuje około stu zwrotów. Potrafi on otworzyć lodówkę, wyjąć butelkę, nalać zawartość do szklanki i podać ją pacjentowi do ust. Wybrani pacjenci mają takie aparaty w domu dla eksperymentalnego sprawdzenia ich sprawności funkcjonalnej Przewiduje się wyposażenie robotów w czujniki optyczne, umożliwiające samodzielne odnajdywanie przedmiotów w innym pomieszczeniu.

Zmienił się też bardzo nawet tak dawno wynaleziony sprzęt rehabilitacyjny, jakim jest wózek inwalidzki. Produkuje się już fotele na kółkach z możliwością ruchu we wszystkich kierunkach i zawracania w miejscu. Najciężej poszkodowani będą mieli ułatwione życie dzięki „inteligentym wózkom inwalidzkim", które omijają przeszkody i w dużej mierze same odnajdują właściwą drogę. Ale i to nie oznacza, że stare sprawdzone metody stają się zbędne: w większości przypadków najlepszy jest jednak wózek inwalidzki o napędzie ręcznym, ponieważ pobudza krążenie!

Opracowała:
KATARZYNA ISAAK

KONFERENCJE

Komisja Informatyki ZG PTE organizuje w kwietniu 1985 r. w Warszawie III Krajową konferencję „INFORMATYKA NTRZĘDZIEM EKONOMISTY". Celem konferencji jest upowszechnienie metod i narzędzi informatyki w środowisku ekonomistów. Tematyką wiodącą jest oferta i zasady działania państwowych, spółdzielczych i prywatnych firm informatycznych. Podczas konferencji odbędą się pokazy działania sprzętu mikrokomputerowego oraz oprogramowania wspomagającego pracę ekonomisty.

Organizatorzy oczekują zgłoszeń komunikatów dotyczących nowych rozwiązań systemowych i oprogramowania wspomagającego pracę ekonomisty. Zgłoszenia należy nadsyłać w terminie do 15 listopada br. pod adresem: Polskie Towarzystwo Ekonomiczne, Zarząd Główny, Komisja Informatyki, 00-042 Warszawa, Nowy Świat 49. Dodatkowe informacje: tel. 27-99-06.

MACINTOSH

Czytając o nowych komputerach w INFORMATYCE odnoszę zazwyczaj wrażenie, że są to „bajki o żelaznym wilku", a w najlepszym razie — opowieści science-fiction. Przedstawiając komputer MACINTOSH firmy APPLE spróbuję więc spojrzeć i z naszej perspektywy.

Zacznijmy od ceny. Otóż kompletny system wyposażony w klawiaturę, monitor, jednostkę dysków oraz drukarkę, a także oprogramowanie — można od kwietnia 1984 nabyć (jednocześnie na całym świecie, co jest nowością marketingową) za ok. 3 i pół tysiąca dolarów. Tu przeciwnicy mikroinformatyki zakrzykną żwawo: „Wariat, oferuje nam komputer za ponad dwa miliony złotych!". Rzeczywiście suma ta wygląda astronomicznie, ale... Po pierwsze, MACINTOSH jest komputerem osobistym drugiej generacji, a w Polsce tak naprawdę nie mamy jeszcze pierwszej (nie licząc zabawek typu SINCLAIR ZX81 — ci, którzy je kupili już wiedzą dlaczego są to zabawki!). Po drugie — i to jest najważniejszy argument „za" — kwota ponad trzech tysięcy dolarów bynajmniej nie odstręcza wielu ludzi choćby od kupienia samochodu Polonez. Wobec racjonowania benzyny oraz znanej trwałości krajowego wyrobu (optymiści twierdzą, że pięć, pesymiści — że trzy lata) zakup POLONEZA wydaje się równie, a może nawet bardziej irracjonalny.

Myślę, że ci spośród czytelników, którzy dysponują wystarczającą kwotą, by porównanie z Polonezem było realne, skorzystają z mojej rady i kupią sobie MACINTOSHA. Życzę im przyjemnej zabawy! Wszystkich innych przepraszam za rozbudzenie apetytów konsumpcyjnych, ale nawet w kryzysie marzenia nic nie kosztują...

Dlaczego MACINTOSH jest komputerem osobistym drugiej generacji? Odpowiedź jest prosta: komputer ten został wyposażony w zintegrowany system oprogramowania (podobny nieco do systemu LISA), który umożliwia pracę z komputerem bez żadnego podręcznika, a nawet instrukcji obsługi. A więc mamy wreszcie do czynienia z maszyną dla ludzi — czyli dla tych co mają do rozwiązania problemy i nie interesuje ich programowanie, a tym bardziej — samodzielne konstruowanie komputera. Wydaje się, że era hobbystów, którzy sami robili sobie komputery, a potem starali się z nich wyciśnąć siódme poty, już się kończy. By wrócić do analogii motoryzacyjnej: dziś nikt (poza maniakami) nie buduje własnego samochodu, choć wielu przystosowuje posiadany samochód do swoich, specjalnych celów. Podobny system zaczyna rządzić światowym rynkiem komputerów osobistych. Następnym krokiem będzie zapewne pełna unifikacja — gdy wszystkie komputery będą używały tych samych systemów operacyjnych, dyskietek i sprzęgów. MACINTOSH jest komputerem zintegrowanym w tym sensie, że użytkownik nie musi już, jak w przypadku poprzednich wyrobów firmy APPLE, kupować kolejnych płytek chcąc przyłączyć kolorowy monitor lub nową drukarkę.

MACINTOSH został fabrycznie wyposażony w możliwość przyłączenia drukarki (Imagewriter, cena ok. 500 dol.) oraz zewnętrznego napędu dysku sztywnego (może nim być Winchester ProFile — ok. 2,5 tysięcy dol. lub też napęd dysków elastycznych — cena ok. 400 dol. — o średnicy 3,5 cala, podobny do napędu wmontowanego na stałe w korpus komputera). Nowy standard dyskietek 3,5 cala opracowany przez firmę SONY, a stosowany już przez wielu producentów (np. HEWLETT-PACKARD w modelu 150), jest bardzo poręczny (średnica dyskietki w sztywnej kopercie-obudowie wynosi tylko 9 cm). Specjalna konstrukcja klapki odsłaniającej materiał magnetyczny dopiero we wnętrzu napędu zapobiega kurzeniu i brudzeniu się dyskietki. Standardowe wyposażenie MACINTOSHA w napęd dyskietek nieco podnosi jego cenę (por. APPLE II). Nowe dyskietki 3,5-calowe mają pojemność 400 KB.

Integracja systemu MACINTOSH polega też na wyposażeniu komputera w olbrzymią pamięć ROM 64 KB, zawierającą cały system operacyjny MAC (dla porównania — APPLE II miał tylko 2 KB ROM, a IBM PC — ok. 8 KB). Dzięki temu MACINTOSH, podobnie jak LISA, ma standardowo myszkę — czyli mechaniczny manipulator kursora. Podobnie też jak w LISIE, myszką wybiera się odpowiednie obrazki (ikony), a maszyna sama wykonuje serie skomplikowanych operacji (jak np. utworzenie pakietu informacji i przesłanie go między dyskietką i drukarką).

Dzięki zastosowaniu procesora MC68000 firmy MOTOROLA (znowu element wspólny z LISĄ) możliwość adresowania pamięci przez MACINTOSHA przekracza 1 MB. W wersji podstawowej komputer jest wyposażony w 128 KB pamięci RAM, ale mówi się o wersji 512 KB. Jedna jedyna płytka drukowana o wymiarach ok. 25 na 25 cm mieści całą pamięć oraz system operacyjny i graficzny. Mieści też osobny zegar, do którego przewidziano bateryjkę, dzięki czemu MACINTOSH nawet po wyłączeniu z sieci może pamiętać czas oraz datę. Ponieważ cały komputer waży niewiele ponad 10 kg, łatwo go przenosić

By ułatwić sprzedaż MACINTOSHA na całym świecie, producent przewidział szeroki zakres dostępnych częstotliwości prądu (od 48 do 68 Herzów) oraz możliwość przełączenia napięcia zasilającego między 110 i 220 V. Ponieważ zużywana moc nie przekracza 60 W, więc nie ma potrzeby wyposażenia MACINTOSHA w wentylator; po prostu płytkę drukowaną umieszczono na dnie obudowy, a zasilacz lampy monitora wysoko ponad nią, uzyskując wymuszony przepływ powietrza przez obudowę. Monitor jest niewielki (przekątna 9 cali), ale o dużej rozdzielczości (512 × 342 punkty) i wyświetla czarne litery na niebieskim tle.

MACINTOSH może generować różne dźwięki, wykorzystując skomplikowany wzmacniacz. Przewiduje się w przyszłości udzielanie odpowiedzi przez komputer ludzkim głosem. Ta innowacja jeszcze bardziej przybliży maszynę nieprofesjonalnemu użytkownikowi.

Komputer wyposażono w zestaw pakietów: MacPaint, MacWrite, MacProjekt, MacTerminal oraz MacDraw. Są one w użytkowaniu analogiczne do opisanych już w INFORMATYCE programów LISY (p. nr 11, 1983), a ich nazwy same tłumaczą zastosowanie. Należy zauważyć, że drukarka Imagewriter pozwala na drukowanie tekstu opracowanego za pomocą MacWrite (różne rodzaje czcionki!) wraz z włączonymi rysunkami wykonanymi przez MacDraw. To też charakterystyczna cecha zintegrowanego oprogramowania.

Zintegrowanie dotyczy też możliwości łączenia wielu MACINTOSHY w sieć. Sprzęt typu RS-422 APPLE BUS pozwala połączyć do 16 różnych komputerów firmy APPLE tworzących lokalną sieć, np. w biurze lub laboratorium. Wspólne wykorzystanie urządzeń peryferyjnych obniża całkowity koszt jednego stanowiska, a jednocześnie ułatwia przesyłanie informacji (teksty, rysunki) oraz współpracę wielu osób.

Chyba najmniej zachwytu budzi zewnętrzna forma plastyczna MACINTOSHA, niektórzy uważają kształt obudowy za nieforemny i niekształtny. Cóż, de gustibus... O tym czy MACINTOSH przyjmie się na rynku, dowiemy się już za kilka miesięcy (podobno LISA nie przyjęła się: w ciągu sześciu miesięcy sprzedano tylko 15 tysięcy egzemplarzy). Eksperci rynkowi uważają że niepowodzenie MACINTOSHA może spowodować bankructwo firmy APPLE. Szkoda byłoby komputerów osobistych drugiej generacji — bankructwo firmy APPLE utwierdziłoby IBM jako jedynego liczącego się producenta komputerów osobistych. A żaden monopol nie jest dobry dla postępu.

JAKUB TATARKIEWICZ

Liczyć lepiej

W amerykańskim tygodniku „Newsweek" ukazała się notatka redakcyjna o panującej w Japonii modzie na posługiwanie się liczydłami. Notatka nosi tytuł, który w wolnym przekładzie brzmi: „Liczydła — groźny japoński zapaśnik". Mieszkańcy Nipponu coraz tłumniej garną się bowiem do zdobycia kunsztu posługiwania się tradycyjnymi liczydłami zwanymi „soroban". W całym kraju prosperuje około 30 tysięcy „jukus", czyli przyspieszonych kursów używania liczydeł, a dziesiątki firm zasypują swoich pracowników instrukcjami do „soroban", żeby odciągnąć ich od elektronicznych kalkulatorków.

„Wyraźnie widać, że teraz zdarza się u nas dużo mniej zwykłych, rachunkowych błędów, niż dawniej" — stwierdza Yukio Tagushi, główny księgowy towarzystwa ubezpieczeń na życie SUMITOMO. Chociaż sposób posługiwania się liczydłami może wydawać się skomplikowany, to doświadczony wirtuoz potrafi na swym „soroban" przesuwać krążki o wiele szybciej niż przeważająca większość ludzi posługuje się klawiaturą kalkulatora. Ostatni zwycięzcy dorocznych ogólnokrajowych mistrzostw „soroban" mogli w ciągu pięciu minut rozwiązać 20 zadań, z kórych każde obejmowało dodanie 20 jedenastocyfrowych liczb. Zalety liczydeł, to nie tylko szybkość — podkreśla Isao Araki, prezes Ogólnojaponskiego Towarzystwa „Soroban" — „Kiedy rachujesz na liczydłach, pozwalają ci one zrozumieć liczby i mechanizm liczenia, podczas gdy kalkulatorek elektroniczny daje tylko odpowiedzi. Ułatwia ci pracę. Ale zrozumieć nie pozwala".

Jednym z głównych propagatorów liczydeł jest koncern MATSUSHITA ELECTRIC INDUSTRIAL, należący do największych wytwórców sprzętu elektronicznego — w tym również kalkulatorów — dla indywidualnych użytkowników. Członek dyrekcji koncernu, Hajim Suzuki stwierdza: „Może to śmieszne, że propagujemy „soroban", podczas gdy wytwarzamy kalkulatory. Ale myślimy, że oba te urządzenia będą nadal współistniały. Przecież nigdy nie przestaniemy chodzić, mimo że są samochody i samoloty". Jak gdyby dla potwierdzenia tych słów firma SHARP, jedna z największych rywalek MATSUSHITA, wypuściła na rynek kalkulator zamontowany na liczydłach, co pozwala użytkownikom przerzucać się od jednego sposobu liczenia do drugiego.

I — jak zawsze — Japończycy próbują eksportować swój ostatni krzyk techniki, czy raczej — rozwiązanie, które na nowo odkryli. Toshio Nakano — człowiek, do którego należą 32 szkoły „soroban" w Tokio i w okolicy — zabrał 20 wykładowców i 47 najlepszych uczniów w podróż do Europy, żeby tam pokazać jak bardzo praktyczne mogą być liczydła. Nakano twierdzi, że jego celem jest udowodnienie iż „czym dżudo w sporcie, tym w rachowaniu może stać się soroban".

(ki)

AFCET

AFCET (Association Française pour la Cybernétique Économique et Technique — Francuskie Stowarzyszenie Cybernetyki Ekonomicznej i Technicznej) jest niedochodową organizacją naukową w dziedzinie informatyki, automatycznego sterowania i badań operacyjnych. Grupuje ok. 4 tys. członków indywidualnych i zbiorowych wywodzących się z jednostek badawczych, administracji państwowej, przemysłu, uczelni itd. Stowarzyszenie stawia sobie za cel rozwój nauki i techniki w zarządzaniu i kierowaniu złożonymi organizacjami. Skupia ono ekspertów z tej dziedziny i wspomaga rozwój krajowych i zagranicznych kontaktów naukowców francuskich, współpracując ściśle z takimi organizacjami zagranicznymi i międzynarodowymi, jak: ACM, IAPR, IEEE, IFIP, IMACS, IMEKO. Bazę działalności AFCET stanowi ponad 100 grup roboczych, zorganizowanych w działy związane z następującymi dziedzinami: sterowaniem automatycznym, automatyzacją prac biurowych, zarządzaniem, komputeryzacją, wspomaganiem decyzji, przetwarzaniem danych, matematyką stosowaną i zastosowaniem systemowym.

Jak większość tego typu stowarzyszeń, AFCET realizuje swoje cele poprzez:

- **wydawanie czasopism specjalistycznych i monografii**
- **organizowanie lokalnych, ogólnokrajowych i międzynarodowych konferencji, seminariów i kongresów**
- **reprezentowanie kraju w międzynarodowych organizacjach informatycznych.**

Stowarzyszenie wydaje następujące czasopisma:

— AFCET Interfaces
— TSI: Technique et Science Informatiques (również w języku angielskim)
— RAIRO Automatique
— RAIRO Analyse Numérique
— RAIRO Recherche Opérationnelle
— RAIRO Informatique Théorique
— Le Nouvel Automatisme

oraz biuletyny, monografie i prace konferencyjne. Członkowie otrzymują bezpłatnie czasopismo AFCET INTERFACES oraz rocznik AFCET — wydawnictwo typu „who is who", zawierające dane o wszystkich członkach Stowarzyszenia (specjalności, kontakty). Indywidualne składki członkowskie wynosiły w roku 1983 — 130 franków (członkowie w okresie pierwszych dwóch lat i korespondenci) albo 260 franków (członkowie zwyczajni).

AFCET
156, boulevard Péreire
75017 Paris
Francja
telefon (1) 766.24.19
teleks EUROTEL 290163 code 235

Oprac. MkS

■ **Przedstawiciele kierownictwa chińskiego przemysłu komputerowego zawarli ze znaną brytyjską firmą SINCLAIR RESEARCH umowę w sprawie podjęcia w ChRL licencyjnej produkcji komputerów osobistych typu ZX 81 oraz ZX SPECTRUM. Umowa przewiduje w początkowym okresie montaż komputerów z części dostarczanych z Wielkiej Brytanii, a następnie sukcesywne uruchamianie własnej produkcji tych części. ZX 81 jest obecnie chyba najprostszym, a jednocześnie najtańszym na rynku światowym komputerem osobistym, ponieważ jego aktualna cena detaliczna wynosi ok. 40 funtów szterlingów (ok. 56 dol.). Plany produkcyjne zakładają zmontowanie w przyszłym roku ok. 30 000 egzemplarzy tego komputera. Należy wyjaśnić, że ChRL uzyskała na zakup sprzętu komputerowego kredyty z Banku Światowego. (K)**

•

■ **Na konferencji prasowej poprzedzającej międzynarodową wystawę i kongres SYSTEMS '83 w Monachium, dyrektor zachodnioniemieckiej filii słynnej firmy konsultacyjnej DIEBOLD udzielił interesujących informacji na temat prognoz rozwoju rynku komputerowego w RFN.**

Oświadczył on, że w całości światowych obrotów tego rynku udział Europy wynosi zaledwie ok. 30%, natomiast RFN — ok. 8%, co odpowiada kwocie ok. 80 mld DM (ok. 30 mld dol.). Podstawowa część tych obrotów, tzn. ok. 56 mld DM (ok. 21 mld dol.) przypada na produkcję sprzętu komputerowego, ok. 18 mld DM (ok. 7 mld dol.) — na przetwarzanie danych oraz ok. 4 mld DM (ok. 1,5 mld dol.) — na technikę biurową. Do 1990 r. przewiduje się stały roczny wzrost obrotów o średnio 7,5—10%. Stosownie do tych prognoz najszybciej będzie wzrastała sprzedaż mikrokomputerów, bo od 30 do 60% oraz terminali (25—50%). Mniejsze przyrosty sprzedaży wystąpią w grupie komputerów biurowych (5—30%) oraz komputerów średnich i dużych (2—20%). Szczególnie dynamiczny rozwój nastąpi w grupie mikrokomputerów, których stan zwiększy się w ciągu najbliższych pięciu lat z 304 tys. egzemplarzy do ok. 2,6 mln. Znaczny przyrost w ciągu tego okresu nastąpi również w grupie terminali ekranowych, których liczba wzrośnie z 31 tys. do ok. 96 tys. Odpowiednio do tych zmian wzrastać będzie również sieć łączności telefonicznej przystosowanej do wymagań zdalnego przetwarzania. (K)

•

■ **Holenderskie stowarzyszenie autorów twórczości muzycznej, zrzeszające kompozytorów, odtwórców i autorów tekstów, postanowiło powierzyć rejestrowanie, wyliczanie i egzekwowanie należności wynikających z przepisów prawa autorskiego, specjalnej organizacji przedstawicielskiej. O ilościowych rozmiarach problemu wyliczania i ściągania należności z tytułu honorariów za publiczne wykonanie utworów muzycznych oraz produkcji nośników (płyt gramofonowych, taśm magnetofonowych, wideokaset) świadczy fakt, że organizacja ta, działająca przecież na terenie niewielkiego kraju, zakupiła do wspomnianych celów aż dwa komputery średniej mocy typu SIEMENS 7.551. (K)**

Tym razem zamieszczamy tekst nie odpowiadający ściśle profilowi tej rubryki. Wacław Iszkowski nie analizuje poszczególnych pojęć terminologicznych, lecz zadaje pytanie o podstawę specyficznego słownictwa w informatyce. Ogólna refleksja powinna ułatwić wybór konkretnych terminów — wprowadzanych do języka polskiego. (Red.)

Nasz drugi język

Informatyk nie znający przynajmniej biernie języka angielskiego jest właściwie analfabetą informatycznym. Pomijam tutaj problemy prezentacji swoich osiągnięć w czasopismach zagranicznych. Pomijam też ograniczony rozwój informatyka nie śledzącego w literaturze zagranicznej najnowszych rozwiązań w dziedzinie, którą się zajmuje. Taki informatyk ma przede wszystkim trudności w przyswojeniu sobie zasad współpracy z dowolnym systemem komputerowym oraz nabyciu umiejętności w sprawnym programowaniu, wykorzystującym dowolny z języków programowania.

Oczywiście, większość informatyków lepiej lub gorzej zna angielski. Pozostaje jednak nie zawsze uświadomiona bariera naturalnej akceptacji znaczenia dyrektyw i zdań programów — wykorzystywanych do komunikacji z komputerem. Bariera ta nie występuje u osób, dla których język angielski jest językiem ojczystym (ang. mother language), znanym od dziecka. Problem ten zilustruję przykładem sesji wykorzystującej język polski do zapisu dyrektyw systemu i instrukcji języka programowania.

```
>jestem Iszkowski
HASŁO=wi
>edycja
SKĄD=terminal
DOKĄD=tekst.źródłowy
* wprowadź
      PROGRAM przykład (wejście, wyjście);
      STAŁE      max=100;
      TYPY       wektor=TABLICA [1..max] Z rzeczywistych
      ZMIENNE    bufor: wektor;
                 i,j,n: całkowite;
      PROCEDURA zamień (ZMIENNE x, y: rzeczywiste);
      ZMIENNE pomocnicza : rzeczywista;
      POCZĄTEK
        pomocnicza:=x;
        x:=y;
        y:=pomocnicza
      KONIEC {zamień};
      POCZĄTEK
        i:=1;
        POWTARZAJ
        wczytaj
        wczytaj (bufor [i]);
        i:=i+1;
      PÓKI (i≤max) LUB (bufor [i-1]<>0.0);
      n:=i-1;
      DLA i:=1 TO n WYKONAJ
      POCZĄTEK
        j:=i+1;
        DOPÓKI j≤ n WYKONAJ
      POCZĄTEK
        JEŻELI bufor [i]> bufor [j]
          WTEDY zamień (bufor [i], bufor [j]);
        j:=j+1
      KONIEC
      KONIEC;
      DLA i:=1 DO n WYKONAJ wypisz (bufor [i])
KONIEC.
* wróć na początek
      PROGRAM przykład (wejście, wyjście):
* znajdź:/wczytaj/
            wczytaj
* usuń
* znajdź : /DLA/
      DLA i:=1 TO n WYKONAJ
* zamień: /TO n/=>/DO n-1/
      DLA i:=1 DO n-1 WYKONAJ
* wróć na początek
* wyprowadź: całość
      .
      .
      . {tutaj wyświetlono na ekranie cały tekst po poprawkach}
      .
      .
* zakończ
KONIEC EDYCJI

>transluj
TRANSLATOR = kopernik
ŹRÓDŁO     = tekst.źródłowy
WYNIK      = tekst.wynikowy
RAPORT     = tekst.raportu

PRZETRANSLOWANO
TRANSLACJA BEZBŁĘDNA

>instaluj
NAZWA PROGRAMU=sortowanie
TREŚĆ PROGRAMU=tekst.wynikowy
DŁUGOŚĆ STOSU=100
ZAINSTALOWANO
>przydziel: wejście<=terminal
>przydziel: wyjście<=drukarka
>uruchom: sortowanie
? 7, 8, 10, 2, 23, 31, 0
WYKONANO
>żegnaj
DZIĘKUJĘ: 15.00 (min.s)
```

Pokazany przykład sesji jest zrozumiały dla nas Polaków, mających podstawowe wiadomości z informatyki. Każda linia tekstu, pomimo istnienia „dziwnych" znaków interpunkcyjnych, jest rozkazem lub krótkim zdaniem opisującym żądanie wykonania pewnej czynności. Również teksty wyprowadzane przez system są zrozumiałe bez potrzeby ich tłumaczenia.

Konstruując ten przykład napotkałem jednak kilka trudności w podaniu zwięzłego i jednoznacznego opisu żądanej czynności. Przykładowo — zamiast rzeczownika edycja lepszy byłby czasownik w drugiej osobie liczby pojedynczej — edytuj, ale takiego słowa nie ma w słowniku języka polskiego. Informacje, których żąda program edytora, są opisane hasłami nie w pełni adekwatnymi do ich charakteru. Zastosowane w przykładzie SKĄD, DOKĄD mogą być źle zrozumiane. Lepszy byłby zapis:

POCHODZENIE_TEKSTU_ŹRÓDŁOWEGO = ...
MIEJSCE_SKŁADANIA_TEKSTU_POPRAWIONEGO = ...

ale są to zbyt długie hasła.

W tekście programu poszczególne słowa kluczowe identyfikują element składni lub czynność programu. Problem pojawia się z odmianą przymiotników identyfikujących typy, jak np.:

x: rzeczywiste oraz TABLICA [1..max] Z rzeczywistych.

Ograniczenie się do jednego przypadku przymiotnika lub rzeczownika wprowadza niegramatyczną postać zdań. Podobne krytyczne rozważania można odnieść do pozostałych części podanego przykładu. Czytelnikowi pozostawiam też zapisanie tej sesji w języku PASCAL oraz w znanym sobie systemie operacyjnym.

Porównując oba przykłady, ten zapisany w języku polskim oraz ten stosowany powszechnie, możemy sobie uświadomić istnienie owej bariery językowej. Dla informatyka pochodzącego spoza obszaru języka angielskiego każda dyrektywa czy słowo kluczowe jest tylko ciągiem znaków i każde musi być albo przyswojone jako pewne hasło lub

dodatkowo tłumaczone na język ojczysty. Istotnym też utrudnieniem jest stosowanie skrótów i akronimów upraszczających komunikację z systemem, ale trudnych do skojarzenia przez „obcokrajowców". Argumentem dla podtrzymania tej tezy niech będzie przypomnienie faktu ostrej krytyki słów kluczowych typu **esac, fi, od, elcyc.** Krytyka ta pochodziła przede wszystkim od urodzonych Anglików i Amerykanów. Dla nich te słowa kluczowe były pierwszymi obcymi, zatem należało się ich pozbyć. Z drugiej strony — rozwijają się prace badawcze poszukujące najwłaściwszych identyfikatorów dyrektyw. Przykładowo — badano zrozumiałość dyrektyw edytorów (**add** czy **append, delete** czy **omit** itp.).

Oczywiście, istnienie bariery językowej nie dotyczy tylko nas. Podobne problemy mają Niemcy, Włosi, Francuzi... Dodatkowe zaś trudności mają Rosjanie i Japończycy, ze względu na odmienny alfabet. Dla nich słowo kluczowe zapisane w alfabecie łacińskim jest tylko pewnym symbolem, tak jak dla nas znak japońskiej katagany. Symbol ten wymaga rozpoznania, a następnie przetłumaczenia, aby można było go zapamiętać kojarząc z określoną czynnością.

Komputery narodziły się w USA, a więc ich językiem ojczystym jest język angielski. Język ten ma wiele cech, które predysponują go do spełnienia roli pośrednika między użytkownikiem a komputerem. Większość słów w tym języku jest krótkich. Szczególnie dotyczy to czasowników określających czynności: wykonaj, zrób, przepisz, usuń, dodaj itp. W języku nie ma odmiany przez przypadki oraz ograniczona jest odmiana przez osoby. Możliwa jest budowa prostych krótkich zdań, po usunięciu przedimków a, an, the.

Ze znanych języków programowania, tylko PROLOG jest oparty na słownictwie francuskim. Język ten stanowi bazę projektu komputerów piątej generacji, realizowanego przez Japończyków. Mało jest jednak prawdopodobne, aby przy tej okazji język francuski (lub japoński) stał się językiem urzędowym maszyn XXI wieku. Zresztą drugi potentat — język ADA — przygotowany również przez Francuzów, podtrzymuje tradycje — używa języka angielskiego i to czasem znacznie przegadanego.

Trudności w nauczaniu i posługiwaniu się „obcym" językiem powodowały inicjowanie prób wykorzystania języka ojczystego w komunikacji z komputerami. W Polsce — na przykład — zaprojektowano język MINI, podobny do FORTRANU. Rosjanie zaś próbowali przetłumaczyć język ALGOL 60 i ALGOL 68 na język rosyjski i wtedy uzyskali poniższe teksty:

```
цел d
если a < b то a: = 0 иначе b: = 0;
для i до n шаг 2 делай
```

Próby te z czasem zostały zarzucone. Wrócono do języka angielskiego, wymagając od informatyków, i nie tylko od nich, jego opanowania. Przyczyna poniechania tych działań, oprócz negatywnych ocen celowości takich przedsięwzięć, leży przede wszystkim w cechach urządzeń zewnętrznych. Repertuar znaków najpopularniejszego kodu ASCII nie zawiera nawet miejsca na wprowadzenie dodatkowych znaków: ż, ć, ź, ą, ę itp., co powoduje, że wiele słów kluczowych stałoby się dziwolągami, jak np.: zlozenie. Rosjanie chcąc przynajmniej opisywać wyniki programów w swoim języku, musieli zrezygnować z repertuaru małych liter. Istnieją też duże trudności technologiczne w adaptacji sprzętu, pochodzącego w wielu przypadkach z krajów zachodnich.

Jakie są jednak konsekwencje takiego stanu rzeczy? Po pierwsze — informatyka jest w tych krajach w pewnym stopniu bardziej hermetyczna, obca dla wielu potencjalnych użytkowników nie znających języka angielskiego. Po drugie — efektywność wykorzystywania komputerów jest niższa, gdyż liczba popełnianych błędów przy wprowadzaniu dyrektyw oraz słów kluczowych języka jest znacznie

→

Udrycki W.: CHILL — język programowania systemów komutacyjnych (I). Powstanie i założenia języka

INFORMATYKA 1984, nr 9, s. 1

Pierwsza część charakterystyki języka wysokiego poziomu CHILL. Omówiono historię powstania tego języka, jego podstawowe założenia, zakres stosowania i ogólną konstrukcję, podkreślając rozwiązania zwiększające niezawodność uruchamiania i eksploatacji programów użytkowych.

Gutman E.: Baza danych obsługiwana przez komputer pośredniczący

INFORMATYKA 1984, nr 9, s. 4

Charakterystyka dotychczas zrealizowanych rozwiązań obsługi bazy danych przez wyspecjalizowany komputer. Omówiono efektywność takich rozwiązań oraz uzasadniono celowość prowadzenia w tym kierunku prac badawczo-rozwojowych.

Zieliński C.: Roboty (4). Złożone języki programowania

INFORMATYKA 1984, nr 9, s. 8

Ostatnia część problematyki współczesnych robotów. Omówiono konstrukcję i działanie specjalizowanych języków wysokiego poziomu, ilustrując przykładami najczęściej stosowanych języków tej kategorii.

Удрыцки В.: CHILL — язык программирования телекоммуникационных систем (I). Возникновение и принципы языка

INFORMATYKA 1984, № 9, стр. 1

Первая часть характеристики языка высокого уровня CHILL. Обсуждено историю возникновения языка, его основные принципы, область применения и общую конструкцию, подчеркивая решения повышающие надежность отладки и эксплуатации пользовательских программ.

Гутман Э.: База данных обслуживанная ЭВМ — посредницей

INFORMATYKA 1984, № 9, стр. 4

Характеристика разработанных до сих пор решений обслуживания базы данных специализированной ЭВМ. Обсуждается эффективность таких решений и обосновывается целесообразность ведения в этом направлении исследовательских работ.

Зелиньски Ц.: Работы (4). Сложные языки программирования

INFORMATYKA 1984, № 9, стр. 8

Последняя часть проблематики современных роботов. Обсуждено конструкцию и действие специализированных языков высокого уровня, иллюстрируя примерами наиболее часто применяемых языков этой категории.

Udrycki W.: CHILL — the programming language for commutation systems (1). The language creation and assumptions

INFORMATYKA 1984, No. 9, p. 1

First part of the CHILL high level language characteristics. History of the language creation, its basic assumptions, application scope and general structure, underlining solutions, which extend reliability of application program's debugging and operation, are discussed.

Gutman E.: Backend data base systems

INFORMATYKA 1984, No. 9, p. 4

Characteristics of realized backend data base systems. Efficiency of such solutions and advisability of research and development in this direction are discussed and reasoned.

Zieliński C.: Robots (4). Complex programming languages

INFORMATYKA 1984, No. 9, p. 8

Last part of modern robot problems presentation. Structure and operation of high level specialized languages, illustrated with examples of the most frequent applied languages, are discussed.

Udrycki W.: CHILL — eine Programmiersprache für Kommutationssysteme (1). Entstehung und Voraussetzungen

INFORMATYKA 1984, Nr. 9, S. 1

Erster Teil einer Charakteristik von der höheren Programmiersprache CHILL. Es wurden Entstehungsgeschichte dieser Sprache, ihre Grundvorausetzungen, Anwendungsbereich und allgemeine Konstruktion mit Betonung von Lösungen, die die Zuverlässigkeit der Fehlerbeseitigung und Betrieb von Anwendungsprogrammen erhöhen, besprochen.

Gutman E.: Datenbasis mit Bedienung durch einen Vermittlungsrechner

INFORMATYKA 1984, Nr. 9, S. 4

Eine Charakteristik von bisher realisierten Lösungen der Datenbasisbedienung mit Hilfe eines spezialisierten Rechners. Es wurden die Effektivität solcher Lösungen, sowie Zwecksmässigkeit von Forschung und Entwicklung in dieser Richtung, besprochen und begründet.

Zieliński C.: Roboter (4). Zusammengesetzte Programmiersprachen

INFORMATYKA 1984, Nr. 9, S. 8

Letzter Teil der Problematik von zeitgenössischen Robotern. Es wurden Konstruktion und Wirkung der spezialisierten höheren Programmiersprachen, illustriert mit Beispielen der am häufigsten angewendeten Sprachen, beschrieben.

większa. Według badań amerykańskich, związanych z projektowaniem analizatorów i weryfikatorów dyrektyw systemu, stwierdzono, że użytkownicy popełniają błędy w 0,554% wprowadzonych symboli, bez uwzględnienia poprawek dokonywanych samodzielnie przez użytkownika. Analizując te wartości należy pamiętać, że piszą oni w swoim języku ojczystym. Ciekawe byłoby więc uzyskanie takich wyników dla użytkowników systemów w Polsce.

Problem ten wydaje się być marginalny, ale trzeba tutaj stwierdzić, że wielokrotnie pomyłki powodują bezużyteczne wykorzystywanie systemu, przedłużają czas jego pracy oraz powodują zniechęcenie użytkowników. Dodatkowe trudności rodzą się przy wykorzystywaniu terminali sprowadzanych z ZSRR o innym układzie klawiszy niż standardowy QWERTY. A trzeba też tutaj dodać, że w USA poszukuje się nowego układu klawiatury, która spełniałaby warunki ergonomiczne dla użytkowników posługujących się dwoma palcami przy wprowadzaniu tekstów w trakcie sesji.

Wróćmy jeszcze do problemu dostępności informatyki dla szerokiej rzeszy użytkowników nieprofesjonalistów. Gwałtowny rozwój jakościowy i ilościowy komputerów osobistych prędzej czy później będzie również oddziaływał na nasz kraj. O nakładach na informatykę w znacznej mierze decyduje zakres zastosowań. Wprowadzenie informatyki „pod strzechy" wymaga przełamania wielu barier psychologicznych. Konieczne jest przekonanie przyszłych użytkowników o celowości stosowania komputera, nauczenie posługiwania się nim i przystosowanie go od strony oprogramowania oraz cech funkcjonalnych do dotychczasowych form pracy. Dla użytkownika posługującego się dotychczas formularzem WZ-62/72 lub kartoteką, jego komputer musi uwzględnić te przyzwyczajenia.

Firma ICL, wprowadzając nowy komputer PERQ z ekranem graficznym, dostarcza oprogramowanie umożliwiające uzyskanie na ekranie dowolnych formularzy lub kartoteki. Użytkownik posługując się myszką steruje kursorem po ekranie wprowadzając z klawiatury dane lub uzyskuje „wyjęcie" żądanej fiszki. To przekonuje dotychczasowych oportunistów do informatyki. A my — nie mając na razie takich atutów — mamy barierę językową.

Jakie są więc propozycje? Nie jest łatwo znaleźć lekarstwo. Kultura informatyczna społeczeństwa jest bardzo niska, a jeszcze przypadek z ZUS-em zniechęca do komputerów. O przyszłości zdecyduje rzetelne nauczanie młodego pokolenia języka angielskiego i prawidłowego wykorzystania komputerów. Starsze pokolenie jest już moim zdaniem stracone.

Komputery osobiste podbijając świat, zmuszają wszystkich użytkowników do nauki języka angielskiego. Po historycznym rozprzestrzenianiu się tego języka w czasach ekspansji Imperium Brytyjskiego, jest to druga młodość tego języka. W rezultacie i dla nas stał się on drugim językiem.

WACŁAW ISZKOWSKI

Poszukujemy oprogramowania użytkowego pomocnego w prowadzeniu przedsiębiorstwa. Interesuje nas możliwość teletransmisji z komputerem ODRA 1305. Odległość od najbliższego ośrodka ZETO wynosi ok. 70 km, linia telekomunikacyjna komutowana.

Oferty prosimy kierować pod adresem:

Lubuskie Zakłady Przemysłu Skórzanego

CARINA

ul. Budziszyńska 4, 66-620 Gubin

Dział Elektronicznego Przetwarzania Danych

tel. 189 wew. 139 lub 103

EO/1098/K/84

amepol

PRZEDSIĘBIORSTWO ZAGRANICZNE W POLSCE

OFERUJE

PÓŁPRZEWODNIKOWE PAMIĘCI OPERACYJNE DO KOMPUTERÓW I MINIKOMPUTERÓW

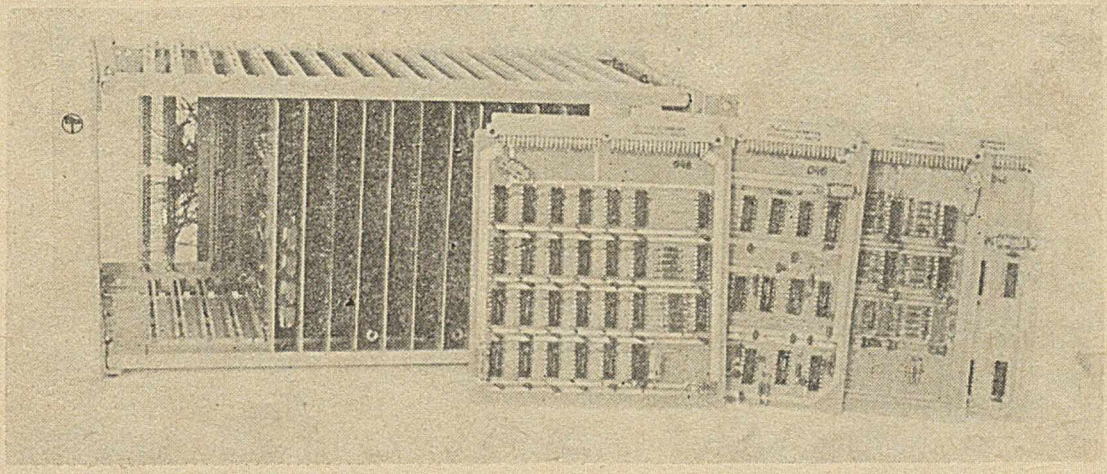
- ODRA 1305 — **o pojemności 64, 128, 192 lub 256 K słów 25-bitowych.**
  Pamięć jest wyposażona we własny interfejs prądowy i może współpracować z pamięcią ferrytową. Moduł o pojemności 256 K słów zbudowany na typowych pakietach ODRA 140 × 150 mm, zajmuje jedną standardową kasetę typu ODRA.

- RIAD R-32 — **o pojemności 256, 512 i 768 KB oraz 1 i 2 MB.**
  Pamięć jest wyposażona we własny interfejs prądowy i może współpracować z pamięcią ferrytową. Słowo składa się z czterech dziewięcio-bitowych bajtów. Moduł o pojemności 2 MB, zbudowany na pakietach Eurokarta 220 × 233 mm, mieści się w jednej standardowej kasecie typu Eurokarta z możliwością jej wbudowania w szafę R-32.

- MERA 400 — **o pojemności od 64 do 512 K słów 17-bitowych.**
  Pamięć, włączana w interfejs M-400, jest zbudowana na standardowych półpakietach 140 × 300 mm i wyposażona w układ akumulatorowego podtrzymania zasilania w przypadku awarii sieci.

- MERA 9150 — **o pojemności 64 K słów 17-bitowych.**
  Pamięć zbudowana jest na jednej standardowej płytce procesora o wymiarach 15 × 15 cali i wyposażona w układ akumulatorowego podtrzymania zasilania w przypadku awarii sieci. Produkowana jest w wersjach umożliwiających zastosowanie w minikomputerach NOVA 1200 lub NOVA 840.

Pamięci produkcji PZ AMEPOL są skonstruowane w oparciu o elementy typu DRAM 64 K × 1 bit przodujących firm zachodnich. Zastosowanie tych elementów zapewnia dużą szybkość działania pamięci oraz znikomy pobór mocy z jednego źródła zasilania.
Ceny oferowanych pamięci są przystępne, a terminy dostawy i instalacji krótkie.



W OPRACOWANIU **PZ AMEPOL** ZNAJDUJĄ SIĘ:

- sterowniki pamięci dyskowej i procesory komunikacyjne do komputerów typu ODRA 1305, MERA 9150 oraz MERA 400
- systemy mikroprocesorowe.

Opracowania te są przewidziane do produkcji w ciągu najbliższych miesięcy.

Prosimy wszystkich zainteresowanych wyrobami informatycznymi naszej produkcji o kontakty listowne, telefoniczne lub teleksowe.

Nasz adres: **PZ AMEPOL**
ZAKŁAD ELEKTRONIKI
00-136 WARSZAWA, Pl. Żelaznej Bramy 1
tel. 20-34-75, 20-45-05, teleks 812539 apol pl

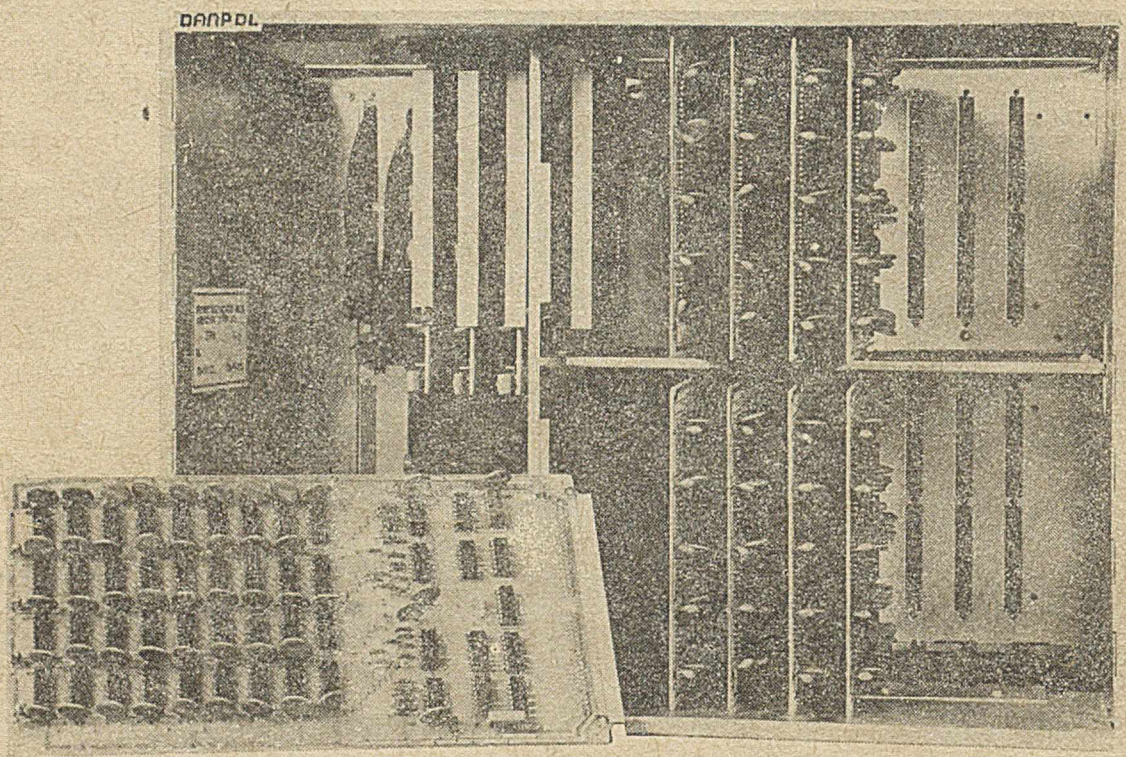
EO/836/K/84

DANPOL

OFERUJE

NIEZAWODNE PÓŁPRZEWODNIKOWE PAMIĘCI OPERACYJNE DO KOMPUTERÓW

- R – 32 o pojemności do 2 Mb
- R – 35 o pojemności do 4 Mb
- ODRA 1305 o pojemności 128 K i 256 K
- PSPD – 90
- PDP – 11
- SM – 4
- ICL – 1903
- ICL – 1904
- ICL – 2903



- OFERUJEMY KRÓTKIE TERMINY DOSTAW
- UDZIELAMY 12-MIESIĘCZNEJ GWARANCJI
- DOSTARCZAMY DOKUMENTACJĘ TECHNICZNĄ
- ZABEZPIECZAMY SERWIS GWARANCYJNY I POGWARANCYJNY
- PAMIĘCI NASZEJ PRODUKCJI PRACUJĄ W PONAD 70 OŚRODKACH W CAŁEJ POLSCE
- NA ŻYCZENIE UDOSTĘPNIAMY LISTĘ REFERENCYJNĄ UŻYTKOWNIKÓW PAMIĘCI

ul. Kopernika 71, 81-456 Gdynia, tel. 22-30-51, tlx 054560

EO/1078/k 84